UNIVERSITY*of*
TASMANIA

Definition and Evaluation of a Conversational Agent System Utilising a Rule-based Knowledge-base System That Maintains Conversational Context

by

David Paul Herbert, B.Sc. Hons

Discipline of Information and Communication Technology
School of Technology, Environments and Design
College of Science and Engineering

Submitted in fulfilment of the requirements for the degree of Doctor of Philosophy

University of Tasmania November, 2020

Declaration of Originality

This thesis is submitted to the University of Tasmania in fulfilment of the requirements for the Doctor of Philosophy. I was primarily responsible for the design, execution, analysis and reporting of the research although it was necessarily undertaken with the assistance of others, who are duly acknowledged in the thesis.

This thesis contains no material which has been accepted for a degree or diploma by the University or any other institution, except by way of background information and duly acknowledged in the thesis, and to the best of my knowledge and belief no material previously published or written by another person except where due acknowledgement is made in the text of the thesis, nor does the thesis contain any material that infringes copyright.

Signed: _____
David Paul Herbert

Date: 13/7/2020

Authority of Access

This thesis may be made available for loan and limited copying in accordance with the *Copyright Act 1968*

Signed: _____
David Paul Herbert

Date: 13/7/2020

Statement of Ethical Conduct

The research associated with this thesis abides by the international and Australian codes on human and animal experimentation, the guidelines by the Australian Government's Office of the Gene Technology Regulator and the rulings of the Safety, Ethics and Institutional Biosafety Committees of the University.

The research has University of Tasmania Ethics Approval: number H0016281

Signed: _____
David Paul Herbert

Date: 13/7/2020

LIST OF RELATED PUBLICATIONS TO THIS RESEARCH

Publications detailing the significant outcomes of this thesis
These papers were first-authored by this thesis author, with the PhD supervisor listed as the second author.

Herbert, D. and Kang, B. H. (2018), Intelligent conversation system using multiple classification ripple down rules and conversational context, *Expert Systems with Applications*, Elsevier, 112, 342 – 352.

Herbert, D. and Kang, B. (2019), Comparative analysis of intelligent personal agent performance, in *Pacific Rim Knowledge Acquisition Workshop*, Springer, pp. 127–141.

Additional publications utilising the C-MCRDR CA system
These papers were not first-authored by this thesis author and were written to present the research related to other fields, however the research further utilises the C-MCRDR conversational agent system as defined and developed in this thesis.

Yang, W., Herbert, D., Kim, S. and Kang, B. (2019), 'MCRDR knowledge-based 3d dialogue simulation in clinical training and assessment', *Journal of medical systems* 43(7), 200.

Gu, X., Herbert, D., Wong, K. C. and Kang, B. (2019), Intelligent web-based task-oriented language assistant using multiple classification ripple down rules, in *17th International Conference on Computers, Communications, and Systems*, Daegu University, Gyeongsan, South Korea.

ACKNOWLEDGEMENTS

I would first like to express my appreciation and respect for my primary supervisor, Professor Byeong Kang for his generous financial support in the form of a full scholarship without which I would have been unable to continue my research, and for the many, many hours of rigorous theoretical discussions about conceptualising ideas and methods, whilst allowing me to leverage his original MCRDR PhD endeavours within my research.

I must also acknowledge the early procedural support and advice provided by Associate Professor Leonie Ellis as well as the authors of some very early associated MCRDR project implementation code by Dr Soyeon (Caren) Han and her husband Mr Hyunsuk (David) Chung.

I'd like to personally thank Associate Professor Kristy de Salas for taking the reigns as a supervisor in the latter part of my candidature – your sage advice and kind assistance have been invaluable in helping me finally wrangle out of the chaos, the thesis document that you now read.

Finally, I would like to thank my long-suffering family, my wife Nicole and son Samuel for putting up with me sequestering myself away to work, helping to maintain my sanity and for their incredible loving support in dealing with my stress and crabby moods and the many, many hours where I could not join in with family activities and family time. Sincere thanks must also go to my cousin Darren and his wife Cathy for scolding and forcing me to '*get on with it*' and focus my energies and attention when I had a tendency to procrastinate and wander.

This research has been partially supported by financial support via a grant from the Asian Office of Aerospace Research and Development (AOARD). The research is also supported by an *Australian Government Research Training Program Scholarship.*

ABSTRACT

Conversational Agent (CA) systems are artificially intelligent software programs using natural language interfaces to simulate real human conversation. They are important as they can provide a human-centric interaction between humans and digital services. This importance is increasing due to the recent, exponential increase of connected devices that provide many disparate services available in homes, businesses and industry. The nature of these services is also changing. Contemporary services that fulfil requests such as web-based information retrieval and setting timer-based reminders are being complemented by new types of services that are emerging due to the enormous expansion of network-connected devices (the Internet of Things). The ability to interact with these services through conversation and modify device behaviour by imparting human knowledge conversationally is a challenge that needs to be addressed for these devices to be truly integrated within society.

The objective of this thesis is to start to address this challenge by defining and evaluating a rule-based CA system that naturally retains *conversational context* and that can be easily maintained by conversational authors who are experts in their own domains. This initial approach is focused to consider domains that do not contain large repositories of existing conversational data for training, and CA systems that do not have a reliance on complex scripting and programming skills or knowledge of formal grammatical syntax that would be needed by the conversational author.

This research is undertaken over two phases. The first phase determines a suitable rule-based knowledge-base system (KBS) methodology subject to conversational and domain constraints that addresses brittleness (lacking knowledge or being unclear as to what knowledge a system contains) which is a common criticism levelled against knowledge-base systems. The Multiple Classification Ripple Down Rules (MCRDR) KBS methodology partially satisfies these requirements and it was adopted as the foundation KBS. This research phase concludes by defining and evaluating Contextual MCRDR (C-MCRDR) through augmentation of MCRDR to facilitate the creation of CA systems for suitable domains. The CA system implemented (C-MCRDR CA) had speech capabilities via specific web browser support, meaning speech input and output were tied to a client computer's speaker and microphone as well as the automatic speech

recognition (ASR) client supported by the browser. For improved versatility and speech-to-text performance, integration of an Intelligent Personal Assistant (IPA) was required. The second phase of the research evaluated properties of two market-leading IPA systems to determine which is the best-performing device to integrate as part of the CA system's user interface. The Google Home and the Amazon Echo were evaluated, with the Google Home demonstrating superior performance. This research phase concluded by defining and evaluating two methods for isolated word transcription error-correction using the Google Home directly coupled to the C-MCRDR CA system.

Significant findings of this research are:

1. C-MCRDR can facilitate an effective, efficient way of maintaining knowledge base systems by acquiring and classifying knowledge incrementally and unlike MCRDR, it maintains conversation context, a factor that assists with allowing utterance classification to change over time as a conversation progresses;

2. Substantial rule-count reduction can be achieved by C-MCRDR (in comparison to standard MCRDR) due to post-inference deferred classifications that include database querying expressions;

3. A natural language interfaces to databases (NLIDB) framework can be engendered by C-MCRDR by way of post-inference query binding and a pattern-matched NL interface with the advantages of knowledge acquisition and maintenance through the ripple-down methodology;

4. Brittleness can be mitigated by C-MCRDR due to the incremental approach to knowledge acquisition, the use of paraphrasal terms in pattern matching, and paraphrasal look-ahead prompting;

5. The Google Home values for the isolated word and phrasal recognition word error rate (WER) measurements were significantly lower than the Amazon Echo;

6. Two error correction schemes have been defined that can be applied to IPAs when coupled to a C-MCRDR CA system to significantly improve the average WER across all datasets.

This doctoral study has made a significant contribution to the body of knowledge through the definition and evaluation of a C-MCRDR CA system. This research demonstrates a contextual rule-based conversational agent system can achieve proven excellent performance results in domains lacking existing corpora of conversational knowledge, while at the same time it

does not encumber conversational authors to require complex scripting or programming skills, or knowledge of formal grammatical syntax. No other human-authored, rule-based contextual conversational agent system based on a derivative of RDR, as opposed to machine learning or statistical approaches, has been detailed and evaluated in the literature. In addition, no other identified studies evaluate and measure the isolated word or sentence-level WER of contemporary IPA devices comprehensively nor do they examine whether attributes of word ranking, and word and sentence lengths, affect the IPA's WER.

The key implication of this study is that C-MCRDR CA systems can perform well without significant technical overheads in their conversational knowledge construction. In domains that do not contain sufficient domain-specific conversational training data, a C-MCRDR CA system may be suitable for the conversational author to build a conversation system that is relevant to their domain. Future work could investigate how such systems can be extended to create a conversational platform that could accommodate many different types of services by extending the conversational knowledge to include behavioural programming, for example, in the form of rule-based actions that are associated with dialog responses.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

> "*Sometimes I think the surest sign that intelligent life exists elsewhere in the universe is that none of it has tried to contact us*"
> – Bill Watterson, cartoonist, "Calvin and Hobbes"

Autonomous and semi-autonomous systems, such as robots and drones, are becoming increasingly popular and in demand in commercial and social environments – deployment of robots will encroach on many facets of human society, sometimes with profound economical and occupational effects (Frey and Osborne, 2017). Robots are also hard to interact with in a natural way if they do not possess a natural language interface (Matuszek, 2015). Humans prefer to communicate using their native language in vocal form, only opting for other methods of communication when necessary (for example, by written forms, gestures and body language) (Barabás et al., 2012). A robot's behaviour is usually prescribed and programmed by its manufacturer for a specific purpose and advanced technical expertise is typically needed to add or refine non-trivial atomic behaviours to adapt to different environments (Suddrey et al., 2017). Creating new behaviours by calling on the *services* of existing atomic behaviours is an exciting possibility. Interacting with robots using conversation and programming by conversation are sought-after goals in artificial intelligence research.

Conversational agent (CA) systems are artificially intelligent software programs using natural language (NL) interfaces to simulate real human conversation (Klopfenstein et al., 2017). CA systems, also known as chatbots, have had widespread commercial success (see Section 2.3.1), and they have recently captured the public's attention, for example, with implementations of Intelligent Personal Assistants (IPAs) such as the Google Home (Google, 2018*a*), Amazon Alexa (Amazon, 2018) and Apple's HomePod (Apple, 2018). The conversational knowledge used by CA systems is broadly sourced through rule-based or statistical machine learning (ML) approaches (Hussain et al., 2019; Mnasri, 2019) (see Section 2.2.8). Knowledge for the rule-

based approach can be captured from experts and represented within a knowledge-base system (KBS). The Multiple Classification Ripple Down Rules (MCRDR) KBS methodology defined by Kang in 1995 is a rule-based KBS that meets the necessary criteria to form the basis for a KBS that could represent the contextual conservational knowledge that is required for robotic interaction (see Section 2.2.5).

As part of a larger research proposal into robotic interaction via natural language speech, this doctoral study was an investigation into the potential use of an augmented version of a MCRDR KBS integrated with a suitable IPA device as an interface to create a CA system as a conversational interaction method. This study identifies and resolves several challenges related to this type of CA system.

1.1 Challenges

CA systems are important as they can provide a human-centric interaction between humans and end-system services, and their importance is growing due to an increasing number of devices (and thus services) being added to the global internet (Adiwardana et al., 2020; Pelk, 2016). The nature of these services is also changing. Contemporary services that fulfil requests such as web-based information retrieval, database access (Grosz et al., 1987; Li et al., 2005), retail shopping (Kasilingam, 2020), and navigation (Apple, 2018) are being complemented by new types of services such as home automation, health monitoring and inventory management. These services are emerging due to the enormous multitudes of network-connected devices (the Internet of Things) that provide many disparate services available in homes, businesses and industry (Casadei et al., 2019). This has led to the broad overarching question – *how can the average person interact with and control these services through conversation*?

The design techniques of CA systems can be broadly categorised as rule-based or statistical machine learning (ML) approaches (Hussain et al., 2019; Mnasri, 2019). Rule-based CA systems use specially-crafted rules to process parsed NL phrases to produce a suitable NL response. These rules can be crafted directly from humans, or generated automatically by ML techniques like rule-based decision trees populated via inductive methods such as Quinlan's C4.5 induction algorithm (Quinlan, 1993), and Induct/RDR (Gaines and Compton, 1995). Statistical ML approaches (such as Recurrent Neural Nets, RNNs, for example, with sequence to sequence (*seq2seq*) learning (Sutskever et al., 2014)), and the inductive methods mentioned above along with their many variants, are trained over existing massive corpora of conversational dialog to create an NL dialog model. In some cases this data is scraped perhaps from domain-relevant social media sites or discussion forums, but the end effect is to predict or classify a category

of CA answer given a user question (Hancock et al., 2019). A problem though is in many domains such relevant conversational training data does not exist and it would take a significant effort for a conversational author (the domain expert) to create sufficient training data. Much commercial and research success has been achieved by human-authored, non-statistical rule-based systems. For example, chemical pathology reporting (PKS, 2020) and pharmaceutical contraindication (Bindoff et al., 2006), see Section 2.2.3). There are also "non AI" rule-based techniques, for example, business process rule management systems (BRMS) such as IBM's iLog (IBM, 2020) and SAS Business Rule Manager (SAS, 2020) which are widely-used, commercially successful mature decision management systems (Zámečníková et al., 2016). Considering this success, the core challenge for this research is: *how can a human-authored rule-based KBS be defined and applied to training-data deficit domains to create a CA system that can achieve high levels of system performance*?

Unfortunately, existing human-authored rule-based CA systems mostly use a syntactically-complex set of conversational scripts to parse NL inputs (Bradeško and Mladenić, 2012), requiring an in-depth knowledge level for the conversational system author. For example, a common approach is to use the *ChatScript* (Wilcox, 2011) or AIML (Artificial Intelligence Markup Language) (Wallace, 2011, 2016) XML-based scripting languages. The famous "ALICE" chatbot and ubiquitous corporate website-style "can I help you?" agent pop-ups follow this style. The problem identified then is that the script authoring approach and its required steep learning curve are not suitable for conversational authors who do not possess an expert level of understanding of programming and formal grammatical syntax. This deficit causes a significant barrier to further conversational knowledge maintenance, as adding further knowledge or correcting mistakes becomes an exercise in complex programming, meaning the knowledge acquisition technique is only suitable for IT professionals. Instead of being able to directly focus on the actual expertise of a conversational author through their domain-specific terminology, language and skillset, the process would be limited by the disconnect or bottleneck of extracting their knowledge through a technical knowledge engineer (Biermann, 1998; Kang et al., 1995). This necessitates an extension to the core challenge for this research: *how can conversational authors specify conversational rule-based knowledge for a CA system when they do not possess complex scripting, programming skills or knowledge of formal grammatical syntax*?

The capture and representation of knowledge from experts as seen in Chapter 2 is a long-studied problem, and knowledge-base systems were the outcome. As discussed in Section 2.2, knowledge-base systems underwent a transformation in the early 1970s by the separation of knowledge (typically represented as rules) from the system program code, instead of this knowledge being implicitly encoded with the program code itself. This separation allowed the knowledge of

the system to be acquired and maintained from the domain expert without the requirement of a system programmer. However, issues still existed as a knowledge engineer was essential to conduct time-consuming, formal interviews with the domain experts to acquire their knowledge and encode the rules that represent it in whatever syntactical and representational format the specific system used. This requirement, and constraints on the domain expert's availability was a significant bottleneck in the traditional knowledge acquisition process. A knowledge engineering acquisition methodology needs to be adopted then that can enable a domain expert to impart their knowledge to the system quickly, directly and easily. MCRDR allows knowledge acquisition quickly and in context without the need for a knowledge engineer (Kang, 1995). With MCRDR as a starting basis for the underlying knowledge representation and acquisition methodology, a related challenge for this research is: *what augmentations to the MCRDR rule-based KBS methodology will produce CA systems that can achieve high levels of system performance*?

Aligned with the issue of knowledge acquisition is the problem of trying to elicit all of a domain expert's knowledge all at once. Experts never justify all of their knowledge, rather they justify why a particular judgement or classification is correct in the specific context in which it arises (Compton et al., 1992). Requiring a conversational author to define all of the conversational knowledge rules at once either programmatically through a script or via other means is clearly a problem. In addition, normal human conversation maintains topical context – previous statements are assumed as a basis of continuity until a topic is changed (Adams and Martell, 2008). With MCRDR as a starting basis, a further related challenge for this research is: *how can conversational context be maintained in a human-authored rule-based KBS*?

Considering the philosophy of conversational authors not requiring high construction overheads in terms of time, grammatical syntactic knowledge and technical expertise, a common approach in CA systems is to utilise pattern matching techniques to process NL inputs (Peng et al., 2020; Bradeško and Mladenić, 2012). Pattern-matched terms are then typically processed by complex scripting languages to produce outputs. A related challenge for this research is: *what is a pattern-matching approach that does not require complex scripting, programming or knowledge of formal grammatical syntax that can provide high levels of system performance*?

Speech-to-text via Automatic Speech Recognition (ASR) is a well-studied problem (see Section 2.3.4) that is enjoying considerable success in personal agents and commercial hardware devices in the last several years (Montenegro et al., 2019). IPAs leverage deep learning and massive data training sets to achieve extremely high ASR performance rates (de Barcelos Silva et al., 2020), and consequently they may naturally lend themselves to be utilised as speech-to-text and text-to-speech components in CA systems. Defining the conversational knowledge

these IPA systems use in a dynamic manner is problematic for conversational authors. Complex vendor-specific programming environments lock in and constrain how NL phrases are processed, typically as statically predefined *skills* or *intents*) (Hoy, 2018; de Barcelos Silva et al., 2020). Given the ubiquity of IPA devices, and the growing presence of CA systems in daily life (Lopatovska et al., 2018), a related challenge for this research is: *what is the best current market-leading IPA device to leverage as a speech component of a CA system in terms of ASR performance that allows their default vendor-defined conversational agent to be supplanted*?

There is currently no identified methodology or system that use IPA devices as NL interfaces where their ASR output can be corrected for errors given such systems have opaque "black box" vendor-specific implementations, and their conversational knowledge can be updated dynamically by conversational authors. A related challenge for this research is: *how can the IPA's associated ASR errors when coupled to a human-authored rule-based KBS be corrected*?

1.2 Purpose Of The Thesis

The purpose of this thesis is to contribute to the body of knowledge surrounding rule-based knowledge-base systems as applied to conversational agent systems by addressing the challenges outlined above. The main aims for this thesis are to:

1. Augment the MCRDR KBS methodology with contextual considerations to create a new rule-based KBS methodology called Contextual MCRDR (C-MCRDR) that can be utilised within a CA system;

2. Develop a CA system based on C-MCRDR that can acquire and apply rule-based conversational knowledge from non-experts[1];

3. Determine whether the C-MCRDR CA system results in a well-performing system in terms of the correctness of the system's responses to conversational questions in a target domain as evaluated by the system architect and domain user's feedback;

4. Determine and evaluate methods to mitigate KBS brittleness[2] through the application of C-MCRDR;

5. Evaluate the ASR error rates of market-leading IPA devices and identify attributes of speech inputs that impact the error rates;

[1]The definition of *non-expert* here means the conversational content domain-expert author does not require knowledge of formal grammatical syntax, nor complex scripting or programming skills.

[2]The definition of *brittleness* here means lacking knowledge or being unclear as to what knowledge a system contains.

6. Determine which market-leading IPA device is the best choice based on its ASR performance to use as a speech interface that allows the default vendor-defined CA to be replaced by the developed C-MCRDR CA; and

7. Define and evaluate methods to correct ASR errors from IPA devices when they are coupled to a C-MCRDR CA system.

The outcomes from points 1 – 4 above have been published in a journal paper titled '*Intelligent conversation system using multiple classification ripple down rules and conversational context*' in 2018 (Herbert and Kang, 2018).

The outcomes from points 5 – 7 above have been published in a conference paper titled '*Comparative analysis of intelligent personal agent performance*' in 2019 (Herbert and Kang, 2019).

1.2.1 Research Questions

To achieve the aims for this doctoral study as described in the thesis' purpose, the core research question explored is:

Core RQ: *How can a human-authored rule-based knowledge-base system be defined and applied to create a conversational agent system that can achieve high levels of system performance without relying on complex scripting and programming skills or knowledge of formal grammatical syntax to specify the conversational knowledge?*

To fully evaluate the core research question, the following sub-research questions (SRQ) have been identified:

SRQ1: *How can conversational context be maintained in a human-authored rule-based conversational agent system?*

SRQ2: *What is a pattern-matching approach that does not require complex scripting and programming skills or knowledge of formal grammatical syntax that can provide high levels of system performance?*

SRQ3: *What augmentations to the MCRDR rule-based KBS methodology will produce conversational agent systems that can achieve high levels of system performance?*

SRQ4:

a) *What is the best current market-leading Intelligent Personal Assistant (IPA) to leverage as a speech component of a conversational agent system in terms of ASR performance that allows the default vendor-defined conversational agent to be supplanted?*; and

b) *How can the IPA's associated ASR errors when coupled to a human-authored rule-based KBS be corrected?*

The outcomes of the research and a summary of answers to the research questions can be found in Chapter 5, Section 5.2.

1.3 Thesis Structure

The rest of this thesis is structured in the following manner:

- **Chapter 2 Literature Review** presents the thesis research in context with the current state of the literature.
- **Chapter 3 Methodology** provides a clear methodology of the planning, justification, and process used to undertake the research.
- **Chapter 4 Evaluation and Discussion** includes the study's results and discussion.
- **Chapter 5 Conclusions and Future Work** provides a summary of results and a context for the significance of the outcomes of the study and an indication of possible extensions to the research.

Chapter 2

Literature Review

"*Imagination, not intelligence, made us human.*"
– Terry Pratchett, fantasy author (Pringle and Pratchett, 1998)

2.1 Introduction

This thesis draws on decades of research from distinct subfields in Artificial Intelligence (AI). From a review of the relevant literature that follows, an opportunity has been identified that combines the fields of conversational agents (CA) with constrained natural language (NL) instruction, knowledge-base systems (KBS) and (in possible extension) human-robot interaction (HRI) to achieve a significant contribution to the field. This contribution is in the form of the definition and evaluation of a C-MCRDR CA system that demonstrates a contextual, human-authored rule-based conversation system can achieve proven high-levels of performance. In addition, the methodology allows non-experts who do not require complex scripting and programming skills and knowledge of formal grammatical syntax to be able to specify the conversational knowledge. This knowledge could then be used to control or interact with disparate end-system services, such as those, but not limited to, an autonomous robot. A natural progression then is to review each of these fields, starting with knowledge-base systems, natural language processing (focusing on techniques for conversational agents), and ending with programming robots (which will be one focus of future work as an extension of this thesis).

Brooks (Brooks, 1986) characterises Artificial Intelligence as a field that is intended to make computers do things, that if done by a human, indicates the presence of intelligence. The term itself was introduced by the computer scientist John McCarthy (McCarthy et al., 2006) in a proposal prepared for a research workshop at Dartmouth College in the summer of 1956, where now well-known researchers including Marvin Minsky and Claude Shannon came together to

formalise the research field. Their initial definition of the term agrees with Brooks' assessment, that, philosophical arguments aside, AI is concerned with *making intelligent machines*. What differs in the field is how to represent *knowledge* that is applied by the ability called intelligence. In this thesis a method for domain experts to provide domain-specific conversational knowledge that can then be used by a conversational agent system is defined and evaluated. This conversational knowledge is represented and embedded in a knowledge-base system, so the literature review begins with a discussion of knowledge-base systems.

The rest of this chapter is therefore organised in the following sections:

2.2 Knowledge-base Systems (KBS)

The current state and history of KBSs with an emphasis on determining a suitable KBS in terms of its knowledge acquisition, inference mechanism and data validation techniques that would be suitable for conversational agents will be examined. This determination is also with focus that knowledge will be acquired incrementally and in context from humans in domains where large corpora of conversational knowledge are not readily available. Important aspects to consider are the ease of acquiring knowledge, maintaining knowledge and validating and verifying the underlying system. It will be seen that *Ripple Down Rules* (Compton and Jansen, 1988), and its derivations fulfil these requirements.

2.2.1 KBS History

Research and development of knowledge-base systems (or the synonymous term Expert Systems) started around the late 1950s, for example, Ledley and Lusted (1959) suggested using a computer expert system for medical diagnosis in 1959. The intent of these systems is that domain-specific knowledge can be transferred from a human expert to a computer system so that non-experts can then use the system to request advice, with the system inferring its conclusion and providing an explain mechanism detailing how the conclusion was determined (Liao, 2005). In other words, a KBS is intended to perform tasks that a human expert would do.

KBSs now consist of four main components (Liao, 2005) – a knowledge-base, a knowledge acquisition tool, an inference engine and a user interface. A necessary component of the interface is an explanation facility that is used to justify conclusions drawn from the system (Dhaliwal, 1996). The inference engine takes the available known facts, and according to the embedded knowledge previously obtained, infers new facts.

2.2.1.1 Knowledge Separation

Although taken for granted now in contemporary systems, early systems embedded their knowledge interspersed amongst the inference program's code. In the 1970s, a key design feature introduced was to separate this knowledge from the inference engine into a knowledge-base. This made it easier to maintain the system's knowledge which is then considered to be declarative knowledge (Compton, 2011). This was a very important advance in KBS research. The knowledge-base itself is then the system's repository of knowledge, typically represented as a series of rules and facts, although a frame-based approach linking objects to facts and values was also common (Abraham, 2005).

The inference engine commonly uses two types of methods – backwards and forwards chaining (Abraham, 2005):

- Backwards chaining starts with a conclusion and works backward to establish the supporting facts, in other words, it starts with a goal and tries to determine what conditions will fulfil it.
- Forward chaining starts with facts and works forward to establish a conclusion i.e. using a set of initial conditions, forward chaining will determine all inferences (conclusions) from those conditions.

As an example, consider the rule-based knowledge-base shown in Figure 2.1 (Abraham, 2005):

```
Rule 1: if A and C then Y
Rule 2: if A and X then Z
Rule 3: if B then X
Rule 4: if Z then D
```

Figure 2.1: Example rule-base

We are trying to assert that D is true, given A and B are true.

Forward chaining:

1. First iteration – find rules whose antecedents can be satisfied. Rule 3 is the only rule that can fire. A, B and X are now true;
2. Second iteration – Rule 2 is satisfied, so Z is true. Rule 4 is now satisfied, asserting that D is true (the goal).

Backward chaining:

1. First iteration – find the rule that asserts D is true. Rule 4 asserts D, so we now must prove Z is true;
2. Second iteration – Rule 2 asserts Z, and A is already true, so we must prove X is true;
3. Third iteration – Rule 3 asserts X and B is already true. We can now conclude D is indeed true.

2.2.2 KBS Problems

Knowledge-base systems enjoyed an explosion of interest in the 80s, with systems such as DENDRAL (Lindsay et al., 1980) – which identified the topological structure of organic molecules

based on mass spectrometer readings; MYCIN (see next section); and PROSPECTOR (Duda et al., 1979) (assessing the likelihood of ore deposits) generating research and commercial success.

Coincident with this success was a realisation of the growing issues related to knowledge engineering. Towards the end of the decade, knowledge-base system interest and confidence waned, primarily because of the problems that arose (Compton and Jansen, 1988). Acquiring knowledge from experts proved to be a difficult, time-consuming process and was a major bottleneck in building an expert system. Compton and Jansen (1988) pointed out that clarifying exactly how an expert made their decisions, and what data they used to do so was also extremely difficult. The other issue was one of brittleness (Lenat et al., 1985) – knowledge-bases need to be updated or maintained to ensure their expertise continues to have relevance in their problem domain.

Two early systems that were identified with these issues are MYCIN and XCON.

2.2.2.1 MYCIN

MYCIN (Shortliffe, 1973; Shortliffe et al., 1975) is a significant, early example of a system developed as part of a doctoral dissertation at Stanford University. It was developed to provide physicians with advice about bacterial infectious disease therapy selection, but it was never used in practise due to possible legal ramifications of potential misdiagnosis. MYCIN's knowledge-base consisted of *IF-THEN* rules with associated ad-hoc certainty factors (CF) (an example of *fuzzy logic*) that range from -1.0 (total disbelief) to 1 (total belief). These CFs needed to be carefully obtained from the domain experts – Heckerman (1986) indicated MYCIN's CFs were elicited from experts by vague statements such as "*Given E how strongly do you believe in H?*". This is important as it is an early example of the knowledge acquisition bottleneck, as busy physicians needed to be available for interview and they also needed to robustly justify their choice of certainty for each rule. As MYCIN's rules are represented as arbitrary LISP expressions, and the consequence of a rule can itself be an arbitrary LISP expression, it means clarity and a clear separation of rules from processing logic can be problematic and subsequently rules required careful authoring.

2.2.2.2 XCON

XCON/R1 was an expert system used to assess customer order requirements to configure VAX computers for purchase at Digital Equipment Corp (DEC) (Polit, 1984; Bachant and McDermott, 1984). It was developed at Carnegie-Mellon University in 1979 and it was designed to

ensure customer orders included combinations of components that could interoperate. Previously every order to DEC needed to be scrutinised by technical editors who then had to provide assembly instructions – a significant bottleneck in time and effort. The initial delivered system included 750 rules, but its knowledge was deficient and not able to configure all customer requirements. After a year of training, the in-house DEC maintenance team maintained the system that included 1000 rules – after considerable effort.

By 1984 the system had reached 2500 rules, but rule maintenance required a major investment in expertise and training for the maintenance personnel. The knowledge engineers had to ensure when they added a new rule to the system that it did not conflict or unknowingly override the existing rules. This validation process took considerable time, which now meant the time validating the knowledge-base was appreciable to the time technical editors spent scrutinising interoperability of components prior to the system's deployment. Clearly this was a problem.

2.2.2.3 Knowledge-base Brittleness

A criticism of knowledge-bases is they can be considered *brittle* when its users are unfamiliar with its content and structure (Chaw, 2009) or alternatively, when the knowledge-base is confronted by problems not foreseen by its builders (Lenat et al., 1985). The latter definition of brittleness and its mitigation was one of the driving reasons behind the knowledge acquisition philosophy of the Ripple Down Rules approach (Compton and Jansen, 1988) which is discussed in Section 2.2.3. The removal of the knowledge acquisition bottleneck means rule maintenance is a relatively fast and easy process for the domain expert to extend the knowledge of the system which can be seen as a brittleness mitigation process.

2.2.3 Ripple Down Rules (RDR)

A direct progression of discussing the problems with classic knowledge-base systems is to examine how these issues are addressed through a newer methodology, Ripple Down Rules. The genesis of the Ripple Down Rules approach begins with the experience Compton and Jansen (1988) had with the GARVAN-ES1 system from three decades ago. Although this reference is significantly dated, it highlights the problems encountered in knowledge-base systems up to that point, so a short discussion of this system is warranted:

2.2.3.1 GARVAN-ES1

Compton and Jansen (1988) had extensive experience maintaining the GARVAN-ES1 system which was an expert system introduced by the Garvan Institute of Medical Research in 1984 at St Vincent's Hospital in Sydney, Australia. The system provided interpretations from approximately 60 possible conclusions of reports from a diagnostic laboratory that measured hormone levels of the human thyroid gland (Compton and Jansen, 1988). During the maintenance of the system they discovered that a significant increase in rule size (a doubling) only increased the accuracy a couple of percentage points. Their analysis showed that some rules were discovered to be incorrect requiring further rules to be added to correct the previous misclassifications, requiring a justification of the reasoning of why a rule was incorrect. This lead them to postulate that instead of asking an expert to justify their reasoning process for the entire domain, an expert need only justify the classification of case in the context in which it arises, based on past experience – comparing a case with a previous case's classification and determining the attributes or features of the case that differentiate it. Compton and Jansen (1988) redeveloped the expert system from the ground up to test this new hypothesis – the justification an expert provides for the classification of a case is very accurate in the context it is given; and if its reasons are used directly as rules in the expert system in the same context, then knowledge acquisition is very simple and fast. They called the approach *Ripple Down Rules* (RDR).

The original non-RDR system achieved an apparent 99.7% acceptance rate of its conclusions by experts, but it was hard to maintain. Rule addition or modification required an experienced knowledge engineer to intercede, required (hard to obtain) endocrinologist consultation, and extensive verification and validation was required to ensure the resulting knowledge-based was not compromised.

Compton and Jansen (1988) stated:

> "*Rules added to this [RDR] structure do far less to corrupt the…existing knowledge in an expert system than with a conventionally structured expert system. Secondly, rules added in this fashion can be added far more rapidly than with a conventional system, so that requirements for maintenance become minor irritations rather than major events requiring hours if not days of work per addition to the knowledge-base*".

This redevelopment built the knowledge-base from scratch, presenting 9514 cases to be classified. These cases came from original patient records that had already been classified by experts for the original system. The resulting system included 347 ripple down rules, compared to the conventional original system's 183 rules. Although there was a considerable increase in rules in

the RDR version, there was a rapid increase in the acquisition rate – at the time 10 rules could be added per hour with minimal difficulty, but the original GARVAN-ES1 system often took half a day for a single rule to be added.

2.2.3.2 Single Classification Ripple Down Rules (SRDR)

RDR (also known as SRDR, *Single Classification Ripple Down Rules*) is a rule-based data classification tool and knowledge representation technique that acquires knowledge from human experts incrementally (Compton and Jansen, 1988). It removes the requirement of needing a Knowledge Engineer to extract knowledge from a domain expert to encode as rules in the knowledge-base which is a significant bottleneck in the knowledge acquisition processes (Biermann, 1998; Kang et al., 1995). RDR-based systems achieve this by allowing the domain expert to provide justification of a conclusion in a local context. Cases consisting of attribute-value pairs relevant to the problem domain are categorised by a decision tree that is constructed and refined over time by classifications and rules provided by a domain expert. By its incremental refinement process, RDR recognises the fact that the knowledge within a knowledge-base may not be complete. The methodology provides an *easy way* for domain experts to correct incorrect knowledge or update the system with new knowledge, and it means RDR highly effective in domains where data becomes available over time.

The original RDR structure is a finite binary tree where each node can have two distinct branches to successor nodes, which are called except and if-not. Each node in the tree has a rule in the format of:

`IF cond1 AND cond2 AND…AND condN THEN conclusion`.

Inference is then conducted by examining a case's attributes, evaluating them from the rules starting with the root node, which has a default classification. If the case satisfies the rule conditions of the node, traversal will progress through the *except* branch. If the node's rule conditions aren't satisfied, the *if-not* branch is traversed. A case's classification results from the last node traversed whose rule was successfully fired.

Using the RDR framework, the expert provides rule conditions to categorise input cases, identifying the relevant features of a case that distinguish it from previous cases. Depending on the domain complexity, the number of rules required can be considerable.

As an example, consider the node traversal path for the classification of a case with attributes (A,C,D), with the RDR decision tree as shown in Figure 2.2.

1. Node 1 (root) is satisfied (it is always True). We have a tentative (default) classification of '0', but we must examine the *except* branch.

2. Node 2 is satisfied (C is present), so we override our classification from Node 1 with a new tentative classification of '1', but we must examine the *except* branch.

3. Node 3 is not satisfied (B is not present), so we examine the *if-not* branch.

4. Node 5 is satisfied, so we override our classification from Node 2, with a new tentative classification of '3'. As Node 5 does not include an *except* branch, and the rule was satisfied, we do not further examine the *if-not* branch. The node traversal halts here, thus the case (A,C,D) is classified as '3'.

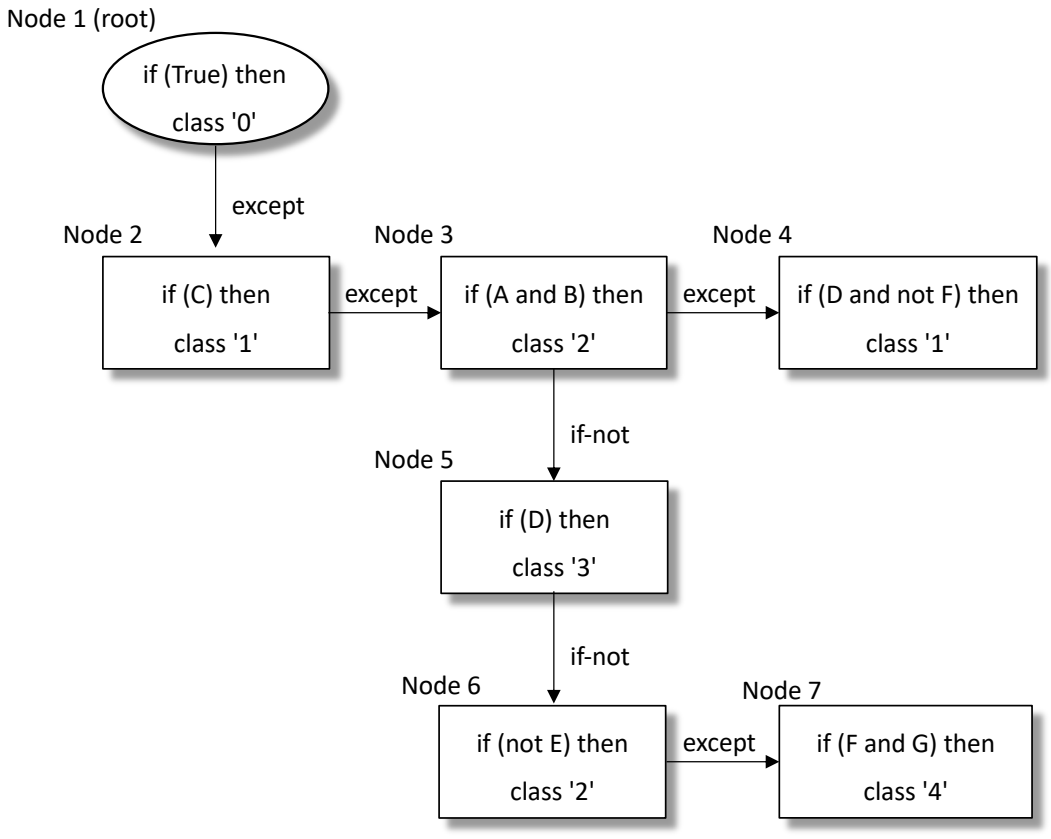


Figure 2.2: Example RDR tree

RDR associates or stores with each node the case that caused the node's creation and rule conditions. This is known as a cornerstone case. During verification of an inferred classification of a new case, if the domain expert disagrees with the case's classification, they are asked to provide the correct classification, and the conditions for a new rule that are used to come to that

conclusion. The conditions of the new rule must differentiate the new case from the cornerstone case in some way in order to override the incorrect classification – assistance is given to the expert in the form of a difference list which simply consists of differences between the new case and the cornerstone case for the last rule that fired. This new rule (node) is added as an exception branch to the rule that concluded with the incorrect classification.

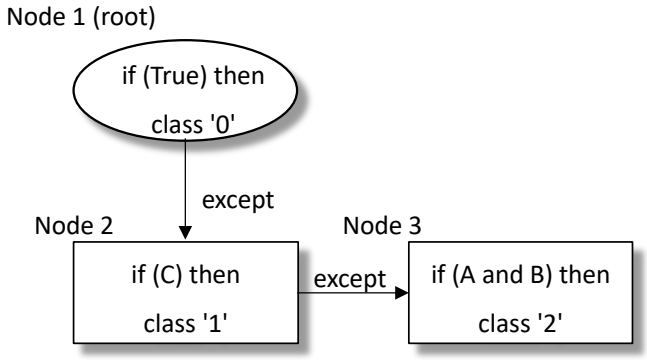


Figure 2.3: Incorrect classification of (A,B,C,D) by Node 3 as determined by expert

For example, consider Figure 2.3 which is an earlier version of the decision tree shown in Figure 2.2 after a smaller number of cases have been classified.

1. Initially, in the present state of Figure 2.3, the new case (A,B,C,D) would be classified as class '2'. The domain expert disagrees with this classification and wants to correct it to be class '1'.

2. Not shown in the figure are the cornerstone cases. Assume Node 3 has the cornerstone case (A,B,C,F) associated with it (this is a previous case that caused Node 3's creation in order to correct Node 2's classification).

3. The difference list between cornerstone case (A,B,C,F) and new case (A,B,C,D) would be (not F, D). The domain expert must choose one (or both) of these conditions to differentiate the new case from the cornerstone case. In the example, they choose both.

4. Node 4 is added to the decision tree, with associated cornerstone case (A,B,C,D) (see Figure 2.4).

2.2.4 Continuing RDR Knowledge-base System Research

RDR's philosophical underpinnings originated in the late 80s, however considerable research and commercialisation of systems has continued to the present day. Ripple Down Rules have

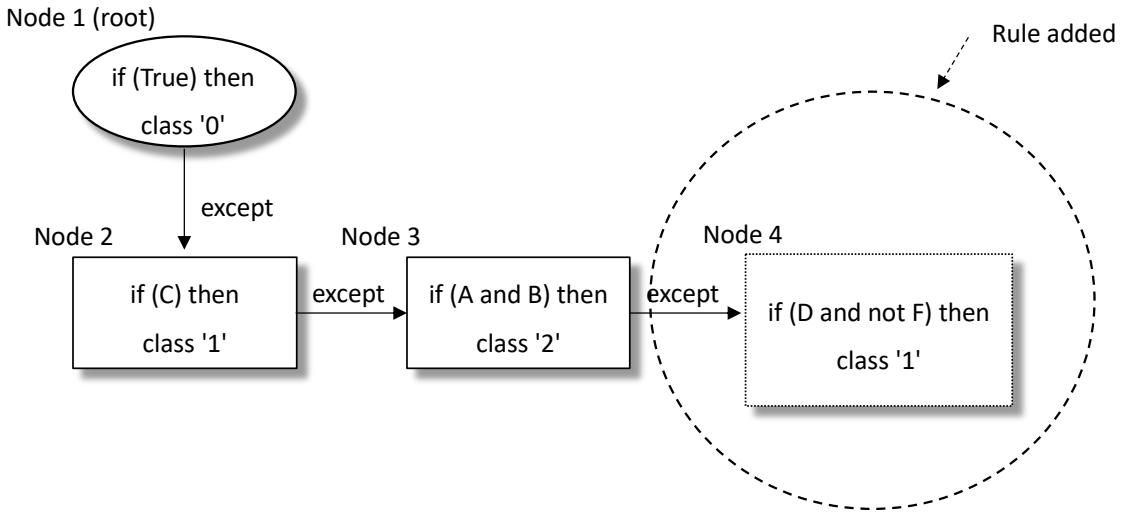


Figure 2.4: Rule insertion

been used successfully for many diverse systems (Compton et al., 2006; Richards, 2009). Examples include health diagnosis and pathology systems (Compton et al., 1992; Han et al., 2013; Compton, 2011; Miranda-Mena, U., Ochoa, Martínez-Béjar, Fernández-Breis and Salinas, 2006), robotic vision systems (Pham and Sammut, 2005; D'Este and Sammut, 2008), robotic control (see Section 2.4.6), morphological, part-of-speech (POS) text tagging for NLP (Nguyen et al., 2016), legal text citation (Galgani et al., 2015), and question and answering (Nguyen et al., 2017; Herbert and Kang, 2018). Repositories of RDR-associated libraries and tools such as the early *Ballarat Incremental Knowledge Engine* (Dazeley et al., 2010) and the contemporary *WEKA Workbench* for machine learning (Frank et al., 2016) enabled many projects that could leverage code libraries without having to develop systems based on RDR from scratch.

Examples of RDR-based health diagnosis and pathology are the early work of the Pathology Expert Interpretive Reporting System (PEIRS) (Edwards et al., 1993; Compton et al., 1992; Preston et al., 1994) and recent commercial products by Pacific Knowledge Systems (Compton et al., 2006; Compton, 2011). PEIRS, like its predecessor GARVAN-ES1 (Compton and Jansen, 1988) (Section 2.2.3.1), was an expert system based on RDR to produce automatic interpretations of pathology reports. Although it is also a considerably dated reference (the system was in operation for approximately four years), pathologists at the St Vincent Hospital in Sydney, Australia were able to build more than 2300 rules without any knowledge engineering or programming support. Pacific Knowledge Systems followed on from PEIRS with the commercialisation of products based on RDR with emphasis on healthcare and interpreting laboratory data.

Question-answering systems are a common theme – Nguyen et al. (2017) use RDR in their question-answering KbQAS system for Vietnamese. Knowledge acquisition via RDR builds a knowledge-base for analysing natural language questions, which produces query tuples as an intermediate representation of input questions. These query tuples are then processed to perform database queries to return a question answer. Input questions are pre-tagged or annotated with part-of-speech (POS) and words are segmented using the Java Annotated Patterns Engine (JAPE), part of the GATE NLP analysis framework (Cunningham et al., 1999, 2002; Thakker et al., 2009). In their earlier work, Nguyen et al. (2016) used RDR to quickly and accurately ascribe part-of-speech (POS) tags to texts from 13 languages in order to facilitate further natural language processing, and unlike other approaches, new exception rules can be quickly added to correct misclassified word tags. Unfortunately the rule knowledge acquisition stages in KbQAS require the domain expert to have significant knowledge and expertise in the JAPE grammar and its annotations.

The LEXA system examines and classifies legal *Distinguished* citations in law cases to ultimately help with court decisions based on prior rulings (Galgani et al., 2015). Like KbQAS, LEXA uses the GATE NLP analysis framework. Their system allows rule re-use, example selection (by suggesting terms to be included in rules) and synonym suggestion (sourced from, for example, WordNet (Miller, 1995)) during knowledge acquisition. The authors report the system outperforms the best machine learning model used and the difference in performance widened when noisier datasets were used.

2.2.5 Multiple Classification Ripple Down Rules

A shortcoming attributed to RDR is that it only provides one classification per case. In some domains, such as interaction with a robot and its sensors, situations may arise when there could be more than one action (classification) for attributes of sensor data (a case) presented to the system. A simple example might be a robot's ultrasonic sensor detects an imminent collision. This sensor data could perhaps result in several behavioural actions, such as reverse, turn left, turn right, speed up, slow down, stop. Alternatively, a conversational agent may need to respond to a user utterance or question with multiple replies, each of which would be a single classification of the source utterance. Consequently, the inference mechanism of the knowledge-base system is required to return more than one classification. Experience from PEIRS (Edwards et al., 1993; Compton et al., 1992; Preston et al., 1994) showed a patient may have more than one disease, and the system coped with such cases by creating a single classification that was in essence a compound disease (Kang et al., 1995). Unfortunately, this can lead to an expansive list of compound disease variations (perhaps differing in the simplest

case by a single disease). Reflections on this limitation by Kang et al. (1995) lead to extending RDR to support multiple classifications for each case presented to the system. This is the intent of MCRDR – it extends RDR to deal with multiple classification while keeping the advantages of RDR as a knowledge acquisition and inference strategy (Kang et al., 1995; Kang, 1995).

2.2.5.1 MCRDR Inference

Unlike RDR, where once a rule is satisfied no rules below it are evaluated, MCRDR evaluates all the rules in the first level of the knowledge-base, and for each rule that was satisfied, it evaluates the child rules, repeating the process until there are no more children to evaluate. This inference process can result in multiple paths, with each path's conclusion being drawn from the last rule satisfied in the path that has no satisfied child rules.

2.2.5.2 Inference Example

An example of an MCRDR knowledge-base is shown in Figure 2.5 (Kang, 1995). Consider the case (a,c,d,e,f,g,h,k) (rules that are satisfied are circled):

1. Rule 0 is always satisfied and if no child rules are satisfied, then the system's default classification is given. As child rules are satisfied here, this classification is not given.
2. Rule 2 (a,c) is satisfied so class 2 is a tentative conclusion.
    (a) Rule 2 has child rules that are evaluated:
        i. Rule 6 (f,e) is satisfied, so class 6 is a tentative conclusion.
            - As Rule 6 has no children, class 6 is a final conclusion.
        ii. Rule 10 (g,h) is satisfied, so class 5 is a tentative conclusion.
            - As Rule 10 has no children, class 5 is a final conclusion.
    (b) Rule 2 has satisfied children, so its conclusion is discarded.
3. Rule 3 (k) is satisfied, so class 2 again is a tentative conclusion.
    (a) As Rule 3 has no children, class 2 is a final conclusion.
4. Rule 5 (d) is satisfied, so class 5 is a tentative conclusion.
    (a) As Rule 5 has no satisfied children, class 5 is a final conclusion.

This example inference shows several paths in the knowledge-base that were traversed to produce the conclusions (Table 2.1). Note paths 2 and 4 both produce the same conclusion, class 5.

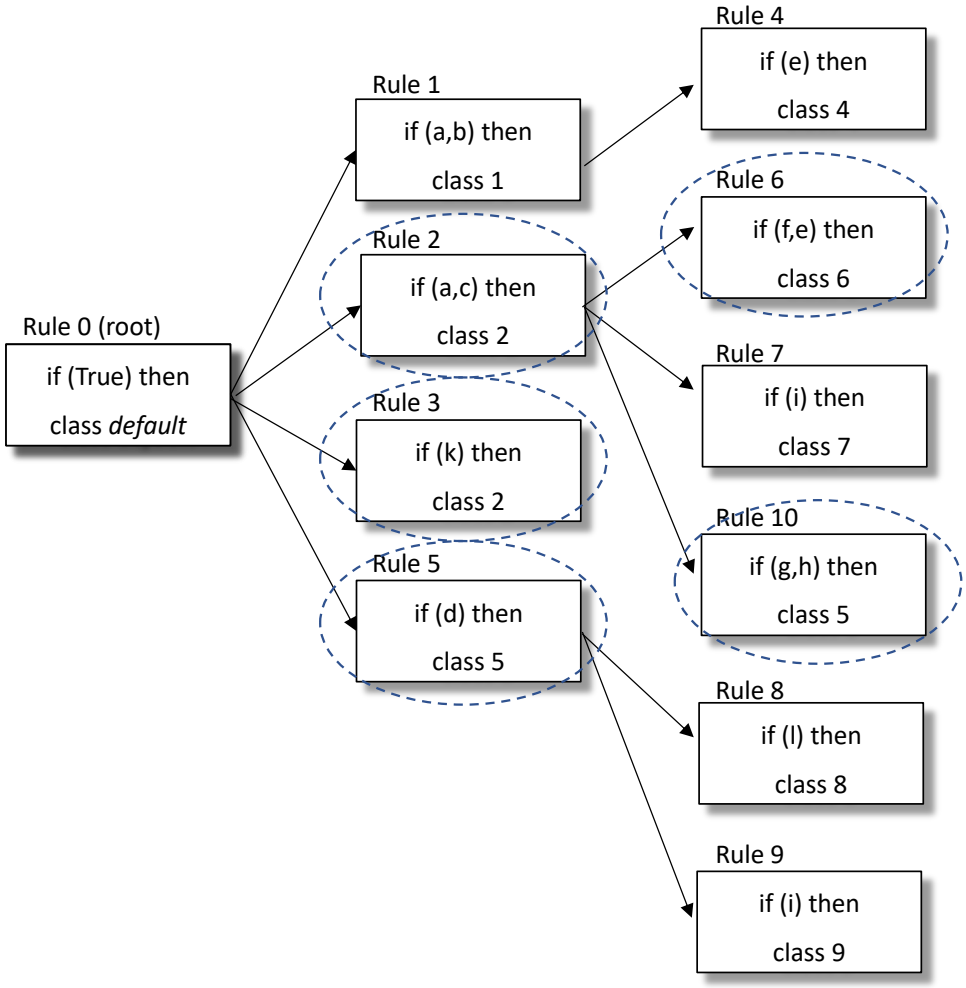


Figure 2.5: Example MCRDR knowledge-base

2.2.5.3 MCRDR Knowledge Acquisition

One minor criticism of MCRDR is that the ease of maintenance is at the expense of the number of rules required to successfully categorise all input cases (Kang, 1995). During inference, a case may be misclassified, or it may not have a classification (if the root classification is ignored). After a classification for the case is obtained via knowledge acquisition from the domain expert, a decision needs to be made as to where the new rule should be added – the rule can be added as a refinement at the end of the path that gave the incorrect classification, or as a new independent classification higher in the tree (and adding a stopping rule that prevents the misclassification).

It is important to note that rules are never modified or deleted – the domain expert refines an existing rule to categorise the current incorrectly-classified case with the addition of an EXCEPT rule, or in some instances, a stopping rule (with no conclusion). This is entirely logical from

Table 2.1: Example MCRDR inference pathways

Path	Rule	Class	Rule	Class	Rule	Class
1	0	*default*	2	2	6	**6**
2	0	*default*	2	2	10	**5**
3	0	*default*	3	**2**		
4	0	*default*	5	**5**		

the domain expert's view, as they do not have to consider issues arising from a reclassification of the current case affecting previous classification rules in the decision tree (Kang et al., 1996). In effect, the addition of a new rule is a localised specialisation of the last rule satisfied in the decision pathway. The decision tree can either become very deep or very wide, based on the system's strategy of deciding during the knowledge acquisition phase where new rules are added – as refinements of existing rules (deeper tree), or at the root as a new independent classification with the old classification stopped by a stopping rule (wider tree) (Kang, 1995) – see Table 2.2.

Table 2.2: Methods for rules to correct a knowledge-base (Kang, 1995)

Requirement	Action to correct the knowledge-base
Wrong classification to be stopped	Add a stopping rule at the end of the path to prevent the classification.
Wrong classification replaced by a new classification	Add a rule at the end of the path to give the new classification.
A new independent classification	Add a rule at a higher level to give the new classification.

2.2.5.4 MCRDR Validation and Verification

Verification and validation (V&V) in essence makes sure a knowledge-base system performs correctly – verification deals with internal consistency, for example, ensuring inference doesn't continuously iterate through a sequence of rules without stopping, and that the knowledge-base system will always reach a conclusion (Preece, 2001); validation tests whether previously (correctly) classified cases will be misclassified by the addition of a new rule (Kang et al., 1995). In other words, validation determines whether the knowledge-base accurately represents the knowledge that was acquired from the human experts.

Preece (2001) states the two processes as:

> "*There is often confusion about the distinction between validation and verification, but the conventional view is that verification is the process of checking whether the software system meets the specified requirements of the users, while validation is the*

process of checking whether the software system meets the actual requirements of the users."

The RDR approach for validation and verification was to simply associate one cornerstone case for each rule as it is created, which results in maintaining a database of representative cases. Batarseh and Gonzalez (2015) classified this validation approach as *test case validation*, and it is the most common validation approach in knowledge-base systems. RDR's rule acquisition process (by constraining the domain expert's rule condition choices for the classification justification of each case) ensures the validity of the new knowledge – the expert is selecting a justification from a difference list between two cases (the cornerstone case and the new case), and the very nature of the knowledge-base structure ensures the knowledge is verified (Kang, 1995).

MCRDR, in order to build on RDR's method, potentially has to consider multiple cornerstone cases as the decision tree can consist of rules being added at the top of the tree, or as deeper refinements of existing rules. This means a number of cornerstone cases can potentially satisfy newly-added rules, so the new rule's antecedent conditions need to be specified (via an altered validation process) to ensure it does not satisfy the previously stored cornerstone cases except for any of those that should, in hindsight, have the same classification as the new rule (Kang et al., 1995). Kang defined an algorithm to construct a difference list for the domain expert to choose from that is formed from attributes of the new case and one of the cornerstone cases that need to be tested – these are determined as those cases that could reach the new rule. Validation occurs by removing all cornerstone cases from consideration that do not satisfy the selected conditions, and then iteratively asking the expert to repeat the process but now with another selected cornerstone case and with the newly-selected conditions being added (conjunctively) to the previously selected conditions. The process stops when sufficient conditions have been added that results in no cornerstone cases left in the consideration list. The algorithm is shown in Figure 2.6.

2.2.5.5 MCRDR Knowledge-base Compression

MCRDR knowledge-bases create larger decision trees compared to other methods (especially inductive methods) as domain experts tend to initially over-generalise their conditions for classification, and then later refine the classification for specialist cases as they arise. This can lead to rule repetition, and the problem of redundant (stopped) rules in the decision tree (Suryanto et al., 1999; Bindoff, 2010). Rule redundancy (in terms of repetition and stopped rules) do not adversely affect the inference process performance, but they hinder or add to the complexity

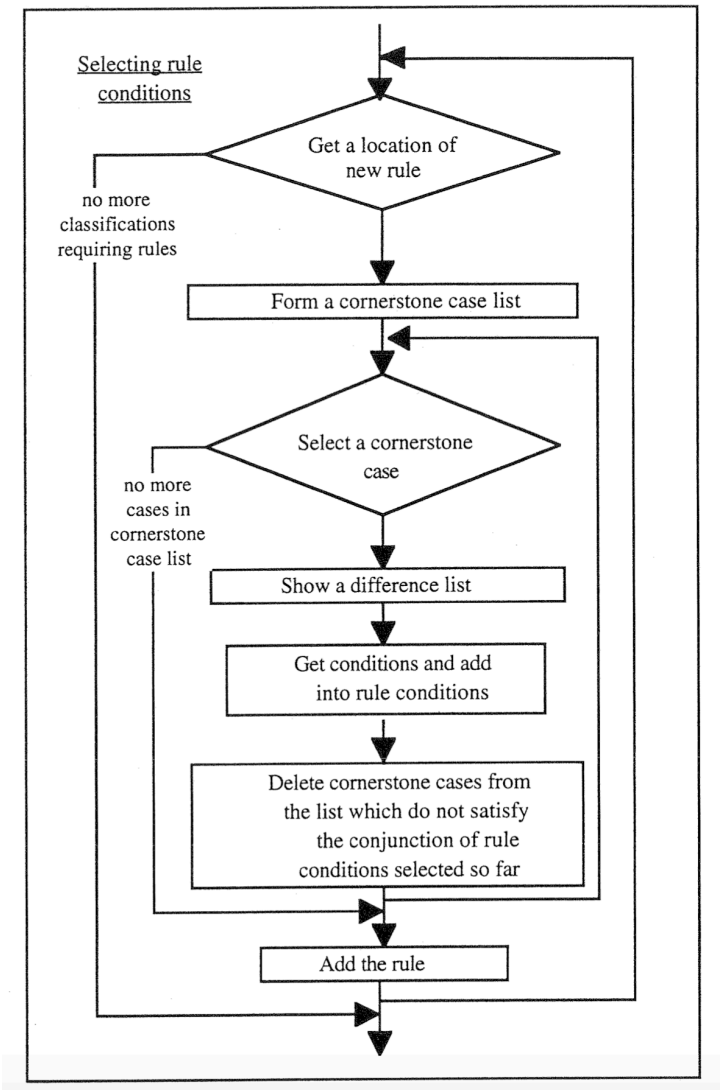


Figure 2.6: MCRDR rule validation (Kang, 1995)

for human interpretation and exploration of the represented knowledge in the decision tree. It must be noted that knowledge-base compression here is referring to the removal or merger of redundant rules that may be repeated in different locations of the knowledge-base (Suryanto et al., 1999).

Although the MCRDR knowledge-base can be compressed, early and comprehensive automatic compression attempts yielded low compression rates (Suryanto et al., 1999). This support arguments against compression, for example, the fact that the associated acquisition history (which may be important in some domains) is lost. The domain expert may also have an expectation (especially during early stages as the knowledge-base's domain coverage is rapidly

increasing) that the decision tree follows their classification justification. In other words, they may have some appreciation of the decision tree structure and they temper their classification responses with this in mind. It should be noted this is against the philosophy of RDR systems as the structure and organisation of the knowledge-base should be abstracted and hidden from the domain expert (Compton and Richards, 1999). Loss of reference to cornerstone cases may also influence validation, indicating compression must be implemented carefully.

Another argument against compression is it may be unnecessary (Kang et al., 1995), as even though the knowledge-base can be 2-4 times in size compared to a knowledge-base produced via machine learning techniques, there is no performance penalty. This may be somewhat anecdotal, as the knowledge-base size is obviously domain-specific, but it is unlikely that a manually-maintained system by a domain expert will produce a knowledge-base containing enough rules that lead to a performance issue during inference on modern day processors. This however may be a factor in less capable hardware, such as lower performance embedded systems, where speed (and storage) are critical.

A form of progressive, or natural, knowledge-base compression, detailed in the first phase of this thesis' methodology, reduces (or minimises) the knowledge-base size through evaluation of an implemented C-MCRDR CA system where generic queries as conclusions are used to leverage context to produce specific results. Although this does not necessarily remove redundant rules that a post-acquisition automatic process would attempt, the knowledge acquisition mindset of the domain expert (by consideration of coincident query construction) may create single rules that have dynamic category-covering conclusions (when bound to specific context) – this results in a smaller knowledge-base that can be argued is a form of compression (please see Sections 3.3.6 and 3.3.7). The added bonus to this method is the knowledge-base is not altered post-acquisition, allowing the domain expert to maintain their knowledge of the structure of the decision tree in situations where this may prove to be beneficial (for example, when examining the reasoning behind an inference result, a domain expert can use this thesis' developed tool to trace satisfied rules in the decision tree (Dhaliwal, 1996) - see Section 3.3.12, Table 3.20).

2.2.6 Continuing MCRDR Knowledge-base System Research

Like its earlier RDR ancestor, MCRDR has been the driving knowledge-base knowledge acquisition and maintenance process behind a considerable breadth and diversity of research. Healthcare is a common field, possibly because many domains already present data in case-based form – some examples are:

- a home-based telehealth diagnostic service through an integrated monitoring and video counselling service touch-screen device (*MediStation*) that uses MCRDR to provide health diagnoses for patients in apartment blocks in Korea (Han et al., 2013, 2015). The MediStation devices allows patients to enter prompted medical data on its touch screen and it also contains biological sensors (for example, temperature and heart rate sensors) and biological sample testing to measure attributes such as protein, pH and glucose levels. All data is returned via a TCP/IP connection to a host server where the collected medical attributes are then used via MCRDR inference to draw possible diagnosis conclusions. The authors simulated the physical MediStation device and developed a proof-of-concept system, however real *in situ* results were not evaluated;

- a pharmaceutical medication review system (Bindoff et al., 2006) that allows pharmacists to review high-risk patient medication and provide decision support (for example, detail medication contraindication). The prototype system described achieved an 80% correct classification accuracy and the validity and usefulness of the approach was proven by the subsequent commercialisation of the decision support system and it is now widely used by pharmacists in Australia (Medscope, 2020);

- a decision support system for detailing breast cancer treatments – an MCRDR-based *Oncology Decision Support Tool* was developed and deployed in a radiotherapy service in a Spanish hospital (Miranda-Mena, Ochoa, Martínez-Béjar, Fernández-Breis and Salinas, 2006). Validation through test case generation resulted either in rule correction, or acceptance by oncologists, and they were satisfied by the system behaviour and the usability of the interface. In addition, usability feedback implied that the oncologists preferred such a system that could provide standard reference treatment suggestions together with their own (rule-refining) in-depth knowledge and experience;

- as indicated in Section 2.2.3, Pacific Knowledge Systems commercialised RDR and MCRDR systems based on chemical pathology (Compton et al., 2006), with the main product, *LabWizard*, which is now known as *RippleDown*. RippleDown is used by 90 pathology laboratories world-wide and performs over one million interpretations per month. The initial client-supplied knowledge-base contains no rules, however the product website states "*Using RippleDown, users can build rules in 1 minute, compared to the industry average of 2-5 rules per day*" (PKS, 2020);

- Torralba-Rodríguez et al. (2010) detail an MCRDR system to classify alarm conditions in an intensive care unit in a Spanish hospital. Patient care attributes such as temperature, heart rate, pH levels, CO_2 partial pressure, diuresis per hour etcetera are evaluated to

produce an alarm that explains the reasoning behind each of the conclusions inferred should an alarm be warranted. The system knowledge-base was incrementally constructed through test cases generated from 43 real critical patients by ICU doctors, and they defined 115 rules. Validation consisted of a Likert-based qualitative survey of 23 independent doctors, which showed high acceptance for usability, but average acceptance for usefulness and slightly above average for accuracy.

Other, non-healthcare applications examples of MCRDR include:

- Industrial sensor and alarm classification for detecting failure in industrial plant (Kim et al., 2018). Based on evaluation at a Hyundi steel plant in South Korea, the MCRDR system categorises and reduces the frequency of equipment alarms so that severe failures will not be missed due to operator overload and "alarm fatigue". The authors collected one year of alarm data from the steel plant and asked 35 experts to classify the alarm data using 48 class attributes. This data was then used by Induct RDR algorithm (see the next section, Section 2.2.7) to pre-populate the knowledge-base – rule maintenance was then conducted to allow experts to correct misclassifications, bringing the accuracy of the system up to 100% for their collected data set;

- help desk systems that use MCRDR to classify troubleshooting knowledge for call centres (Vazey and Richards, 2005; Ho Kang et al., 1997). Keyword searches are used in MCRDR inference to classify (or refine) the search terms to then retrieve associated troubleshooting documents as conclusions. Vazey and Richards (2005) developed a prototype system that enabled call centre staff to find how problems were solved in the past based on attributes recorded in an incoming client issue incident. They can then refine or recategorise the results according to the context of the given problem class if needed by rule maintenance, with conclusions referring to URLs for accessing a defect tracking database;

- the *Sonetto* online grocery shopping system used by Tesco.com (Sarraf and Ellis, 2006; *Sonetto Enhances Online Experience @ tesco.com*, 2005) for business knowledge capture, for example, defining product shipping costs based on attributes such as price and weight, or offering shopping suggestions based on items purchased. Multiple vendors and categories of product are reduced to a single version of product specifications, which are maintained and determined by internal category managers without reliance on IT technicians. Sarraf and Ellis (2006) claim Tesco.com is the most successful online grocer in the world.

2.2.7 Other RDR variations

The benefits in the knowledge engineering approach given by RDR have been further enhanced by continuing variations of the core concepts, typically by modification of the inference mechanism to support intermediate or hierarchical conclusions, which is something RDR and MCRDR lack. Some of the derivations are:

- Induct RDR (Gaines, 1989; Gaines and Compton, 1995) Although not a derivation per-se, Induct RDR combines knowledge creation through a machine-learning technique (rule generation through statistical tests) to produce an RDR knowledge-base from a large dataset that can then be coupled with knowledge acquisition from human experts. Recent examples of research conducted with Induct RDR include a diagnostic reporting system for several medical categories (Hyeon et al., 2016; Kim et al., 2016) (due to the limitation of Induct RDR only supporting single classification, the authors divide their domain into 18 separate categories/domains for diagnosis, and they report the performance as poor and it requires further research); phishing website detection (Chung et al., 2016) that modifies Induct RDR to better support numeric and nominal data types – the authors used 75 datasets from the UCI machine learning repository (Dua and Graff, 2017) to compare their modified Induct RDR algorithm against Ridor (Induct RDR), J48 (C4.5 (Quinlan, 1993)) and NBTree (Naive Bayes) from the Weka suite provided by the University of Waikato, New Zealand (Frank et al., 2016). They found promising results where the algorithm outperformed all others in 30% of the datasets, and it outperformed the "native" implementation of Induct RDR, Ridor, in 66% of the datasets;

- Nested RDR (NRDR) (Beydoun and Hoffmann, 1997, 2001) is an extension of MCRDR that allows intermediate concepts or abstracted terms to be expressed in an expert's own terminology, with each concept defined as a separate RDR tree, producing, in effect, a hierarchical structure. Each defined concept can then be used as a higher-order attribute to define other concepts. Hamade et al. (2010) use NRDR to capture CAD designer's specifications for "dimensional tolerancing", specifically applied to the classical mechanical fit problem between a shaft and a hole. Their system can infer fit schemes (upper and lower bound tolerances) for shafts and holes (for example, whether the fit is to be loose or tight), or alternatively, given the diameters of a shaft and hole, inference can determine the type of fit. It should be noted criticism of NRDR includes "…[the] *hierarchical structure of the knowledge base causes problems for keeping the entire knowledge base consistent when a single concept needs to be altered*" (Richards, 2009; Beydoun and Hoffmann, 1997);

- Recursive RDR (RRDR) (Mulholland et al., 1993, 1996) used inductive methods to define eight RDR knowledge-bases that were associated with each methodological component of ion chromatography in the field of separation science, a configuration task. The final conclusions of each RDR component were then filtered by a heuristic algorithm to determine which conclusions to accept as inputs in each iteration of inference that then allowed the gradual buildup of a final ion chromatographic method (Richards, 2009). The solution however was unreliable due to the fact it could not cope with cyclic knowledge well (Bindoff and Kang, 2011);

- Repeat Inference MCRDR (RIMCRDR) (Compton et al., 1998; Compton and Richards, 1999) like NRDR, RIMCRDR extends MCRDR with the provision for intermediate concepts, but unlike NRDR's repeated refinement process requiring the expert to resolve conflicts, RIMCRDR instead processes rules in a strict chronological order (the order rules were added to the decision tree) – inference is a recursive process that continues adding tentative conclusions and permanent additions to a case based on whether a rule has satisfied children until no further changes are made (Compton and Richards, 2000; Finlayson, 2008). RIMCRDR arose from previous studies in complex configuration tasks, such as the ion chromatography work of Mulholland et al. (Mulholland et al., 1996).

- Dynamic RDR (DRDR) (Shiraz and Sammut, 1997) and Learning Dynamic RDR (LDRDR) (Shirazi and Sammut, 2008) - see also Section 2.4.6; DRDR extended RDR to include multiple knowledge-bases that allow different cognitive processes to be treated separately, for example, allowing a pilot to specify rules in one knowledge-base to adjust a plane's elevators for altitude adjustment, and another to control ailerons. DRDR also allowed rules to be specified through a graphical user interface (that perhaps features relevant controls associated with specific instrumentation), as well as transferral from a paused flight simulation tool. In contrast to the manual rule acquisition process of DRDR, LDRDR uses logged data to automatically create or supplement rules in a current knowledge-base. This allows rules for sub-cognitive data to be captured, for example by a pilot, when they cannot articulate the rules manually but can generate a wealth of data by using instrumentation and controls of a plane.

- Multiple Classification Ripple Round Rules (MCRRR) (Bindoff, 2010) – Bindoff extends MCRDR to include existing classifications in rule conditions, which required careful consideration and provision to detect cyclic rule definitions as the underlying n-ary tree of MCRDR was altered to become a directed graph. The method was successfully evaluated by a complex configuration task (Bindoff and Kang, 2011), however the authors

acknowledge the experiment may suffer from designer bias as they also customised the configuration task (with implied in-depth knowledge of the MCRRR method that other experts would not possess).

2.2.8 Other KBS, ML Methods and RDR Justification

Ripple Down Rules (Compton and Jansen, 1988) (see Section 2.2.3) and all of its further derivations arose from the initial early experiences with GARVAN-ES1, but there are many other techniques for knowledge representation, acquisition and classification. However, RDR (or more precisely, in this thesis, MCRDR) was chosen as it serves as an excellent starting point for augmentation (as detailed in Chapter 3) to engender conversational agent systems based on its knowledge maintenance and acquisition methodologies. Its suitability to domains that do not contain massive corpora (that are needed to train the statistical machine learning approaches, such as seq2seq (Sutskever et al., 2014), or inductive/learning methods (Quinlan, 1993; Gaines and Compton, 1995)), and the requirements detailed in Chapter 1 of a human-maintainable rule-based methodology are precisely matched by RDR.

> "*...machine learning methods have some drawbacks: (1) In order to train [a] classifier, human must collect large number of training text terms, the process is very laborious. If the predefined categories changed, these methods must collect a new set of training text terms. (2) Most of these traditional methods haven't considered the semantic relations between words, so it is difficult to improve the accuracy of these classification methods*" (Khan et al., 2010)

Supervised machine learning's need of a large corpus of pre-labelled (categorised) examples is also supported by Sebastiani (2002) in their survey of text classification techniques.

> "*In the ML approach, the preclassified documents are then the key resource. In the most favorable case, they are already available; ...The less favorable case is when no manually classified documents are available; this typically happens for organizations that start a categorization activity and opt for an automated modality straightaway.*"

This is relevant as the classification of conversational utterances to system responses could be viewed as form of document classification. This model however does not take into account the possibility that the set of all resulting categories (which, in this thesis' context is the conversational reply) in conversation may be considerable for a given domain, and categorisation

determination is also dependent on the current conversational context, which is perhaps also a form of temporal data. For example, the initial categorisation of text utterance $t_1 \rightarrow$ response r_1 may then differ as conversation later progresses, i.e. $t_1 \rightarrow$ response r_2. For example, consider the following conversational snippet:

I do not like brussel sprouts (t_1)
Yes you do! (r_1)
...
I do not like brussel sprouts (t_1)
I agree, they are horrible! (r_2)

Such mutability in the classification based on context is thus also difficult with traditional ML algorithms (Revesz and Triplet, 2011). The entire time-series labelled data can be used as a feature (i.e. with time as an attribute) but this again would require a pre-labelling of a large corpus of data over many sequences of time, which is a huge undertaking in any domain (Fattah et al., 2016).

> "*Data classifiers, such as support vector machines or SVMs, decision trees, or other machine learning algorithms, are widely used. However, they are used to classify data that occur in the same time period*" (Revesz and Triplet, 2011)

These are the justifications for choosing the MCRDR derivative as the KBS approach taken by this thesis. Unlike other work using supervised ML methods based on feature or pattern classification, for example, probabilistic classifiers like Naïve Bayes (Rish et al., 2001; Saritas and Yasar, 2019) or linear/non-linear classifiers like support vector machines (SVM) (Vapnik and Vapnik, 1995; Burges, 1998; Chen et al., 2006), and k-nearest neighbours (k-NN) (Khan et al., 2011), RDR acquires knowledge incrementally as cases are presented to the system, and it uses a human to justify the classification, instead of identifying standout or hypothetical cases through which knowledge will be structured (Richards, 2009). Several theses could be written surveying machine learning classification and knowledge engineering techniques and this is not an aim of this research. It is perhaps prudent though to (briefly) acknowledge other major knowledge-engineering approaches:

- *Case-Based Reasoning (CBR)*
  CBR uses specific knowledge of previous experience (problems). When a new problem arises, it is solved by finding a similar past case and reusing it in the new context.

CBR is also an incremental approach as new cases are retained each time a problem is solved (Aamodt and Plaza, 1994).

- *Formal Concept Analysis (FCA)*
  FCA derives a conceptual hierarchy of objects and their properties. Objects that share the same values for a set of properties are grouped as a concept in the hierarchy, with sub-concepts grouping subsets of the objects from the parent concept (Wille, 1992).

- *Repertory Grids*
  Repertory Grids are designed to "*…give access to a person's underlying construction system*". This is done by identifying the objects in a domain and determining how to distinguish between those objects by requesting the person (the domain expert) to rank the objects against concepts (Gaines and Shaw, 1993).

2.3 Natural Language Processing (NLP)

The discussion now turns from knowledge-base systems as repositories of knowledge and in particular, RDR's methodology to acquire and maintain that knowledge, to consider interacting with systems using natural language. Systems are hard to interact with in a natural way if they do not possess a natural language interface as humans normally communicate using their native language in vocal form, falling to other methods of communication when necessary (e.g. written forms, gestures and body language) (Barabás et al., 2012). Decades of research have been dedicated to this area in many areas of AI, but the focus is narrowed in this section to consider in particular natural language interfaces to databases (NLIDB) and conversational agents (CA) as they are directly relevant to the thesis' research questions.

2.3.1 Conversational Agents

As previously mentioned, this thesis investigates and defines a methodology that can acquire and apply rule-based conversational knowledge for a conversational system by conversational authors without requiring them to have scripting and programming skills, or knowledge of formal grammatical syntax. The developed system creates a form of a chat-based interface that also implicitly maintains topical conversational context (demonstrated in the first phase of the methodology, see Chapter 3).

It is worth noting that usually the main impetus for conversational-based natural language development in contemporary systems is to produce a system that has near-human responses to external user stimuli which can potentially fool a human into believing they are interacting

with another human (Shah et al., 2016; Westerman et al., 2019). This is not an objective of this thesis – at no point is the C-MCRDR CA system expected to achieve a linguistic and semantic level understanding where it could potentially pass a so-called Turing Test (Turing, 1950), or be ranked against populist approaches that use "trickery" to give the illusion of a human behind the agent.

Conversational agents (CA), also known as Human-Computer natural language discourse systems or chatbots, are a prime example of a paradigm in Artificial Intelligence (AI) that has significant public exposure and interest. These systems have seen many fictional depictions in popular entertainment media – examples range from the fully self-aware conversant human-like intelligences such as the apparently malign control intelligence of the Discovery spacecraft, HAL 9000 (Kubrick, 1968) and the eccentric but benign humanoid protocol droid C3PO (Lucas, 1977), to a more traditional dial-up command line interface with the naïve but dangerous nuclear weapons supercomputer controller, War Operation Plan Response (WOPR) (Badham, 1983). These systems, although fictional, project the goal of discourse systems, namely to allow a human to interact freely and naturally using unconstrained language.

Fictional portrayals aside, natural language systems such as conversational agents have been the focus of considerable research in the modern digital era (Mauldin, 1994). A conversational agent is usually a text-based system comprised of a server component and one or more client components, typically communicating over a computer network (Klopfenstein et al., 2017). The most common type takes the form of a question and answer (QA) system, comprised of the server AI backend application, and a dedicated application or browser-based client. Interaction is then a conversation between a human client and the backend system. There is an increasing popularity, ubiquity and public awareness of such systems (Klopfenstein et al., 2017), for example, in the public sector according to Androutsopoulou et al., conversational agents are most prevalent in social media systems such as Twitter, where for example, political agencies attempt to sway public opinion (Androutsopoulou et al., 2019). Mehr et al. (2017) in a Harvard study expand on the increasing use of AI and conversational agents by considering how the public access governmental resources (which are also applicable in the private sector), and they identified common use cases for three (of five) categories that are applicable to conversational agents:

1. *answering questions* – for example, the possible rationalisation of call centre operator times by freeing them up to assist with complex queries, with the simpler queries handled by chatbots;

2. *filling out and searching documents* – for example, auto-completion of governmental forms and assistance with completing subsequent form fields; document search and retrieval from large data archives;

3. *routing requests* – classifying requests and routing the user to the correct office. This is also a typical private sector use case, for example, telephony services utilising speech to text to route a customer query to the appropriate department.

Conversation agents, irrespective of their *embodiment* (web sites, stand-alone software systems, robots, telephony, games, intelligent personal assistants etcetera) are becoming more influential and useful to the general public and are an obvious and visible application of AI (Lopatovska et al., 2018).

2.3.1.1 Conversational Agent Methods

Conversational agent system methods are broadly defined by two major categories: template or patten-matching methods and statistical language models like neural networks (Peng et al., 2020; Griol et al., 2008), with the statistical approach further categorised by task-oriented and open-domain neural models (Budzianowski and Vulić, 2019). Significant results in the statistical approaches for example have been achieved by systems such as GPT-2 (Radford et al., 2019) that has been trained on massive datasets. Consider a simple question-answer example drawn from GPT-2:

> *There are six frogs on a log. Two leave, but three join. The number of frogs on the log is now? Seventeen* (Marcus, 2019)

Even though the current state-of-the-art neural network approach, GPT-2, can generate sensible and comprehensive results from given sentence fragments in open-domains, for example, here responding with a correct category (a number), it shows it does not possess *understanding* of the actual question. Such systems are also difficult to train or maintain by non-experts in machine learning in their specific domains as they require massive corpora to train on (Hancock et al., 2019), and considering the overall research aims and problems being addressed by this thesis (see Section 1.1 and the main research question, Section 1.2.1), a pattern-matching, rule-based method is adopted. RDR, based on properties reviewed in Section 2.2.3, was selected as the underlying methodology and technique for knowledge acquisition and inference. Being an incremental, rule-based approach, it is highly suited to domains without large existing repositories of conversational data. In terms then of NL processing techniques, attributes of

cases (dialog utterances) presented for knowledge acquisition or inference become the key terms that are parsed for classification (the system response) – this is a form of pattern matching. *Understanding* in these forms of pattern-matching systems is thus delegated to the conversational author themselves, which is exhibited by the rules maintained in a knowledge-base or alternatively via syntactical-matching statements present in a script.

More specialised or targeted version of pattern-matching (or "slot-filling") systems are leveraged by the software agents in the Intelligent Personal Assistants category. These systems, now typically included in modern smartphone or speaker operating systems such as Google's Android OS and Home speaker (Google Assistant) (Google, 2018*a*), Apple's iOS and HomePod speaker (Siri) (Apple, 2018), Amazon's Echo speaker (Alexa) (Amazon, 2018) and Microsoft's Windows 10 (Cortana) (Microsoft, 2018), are becoming ubiquitous and familiar to the lay public (Hoy, 2018). They have an Automatic Speech Recognition (ASR) front-end for input and Text To Speech (TTS) (where relevant) for output (Yu and Deng, 2016), but the recognised transcribed text is used as an intermediate form for processing. This category of virtual assistants extends a passive conversational agent dialog paradigm by facilitating voice-activated control along with tight integration with operating system services on their hosting platform (e.g. calendaring, email, messaging etcetera) as well as other curated responses from external web-based services. These are commercial, proprietary systems, and as such, scholarly implementation and technological details are difficult to determine but they are likely to use statistical, or *Deep Learning* (neural nets) as an underlying technology, especially for ASR-related transcription.

2.3.1.1.1 Pattern Matching

The most common implementation approach for conversational agents uses structural pattern matching, typically requiring an author to modify a script that is used in a specific problem domain.

> "*[Pattern Matching] …is by far the most common approach and technique used in chatbots. Variations of some pattern matching algorithm exist in every existing chatbot system*" (Bradeško and Mladenić, 2012).

Pattern matching does not consider the overall semantic or grammatical meaning of input, but in the most basic form, it identifies explicit keywords, perhaps using exact or approximate string matching techniques (as discussed in Section 2.3.3), that result in a specific, pre-defined output as a response. The pattern matching approach can be enhanced with more advanced linguistic analysis techniques, such as Part of Speech (POS) tagging (Manning et al., 2014), or n-gram

matching (Section 2.3.3.2) at the word (and syntactic) level where grammatical classes and morphological information is considered – this however is then approaching syntactic grammar systems (Section 2.3.2.2) in the style of approach.

2.3.1.2 Well-known Historical Pattern Matching Conversational Agents

2.3.1.2.1 Eliza

ELIZA (Weizenbaum, 1966) is often cited in nearly every academic study concerning conversational agents, and it was certainly a system that achieved public awareness. Very early human-computer interactions with natural language began with the ELIZA system which ran on an IBM mainframe 7094 at MIT. ELIZA was designed to act as a simple psychotherapist, responding to user input by pattern-matching decomposition rules that are defined in a separate, external script to the application. These rules are triggered by detection of keywords in the user's input text, and associated with each matched decomposition rule is a list containing one or more reassembly rules that are used to provide a response to the user. A rule is chosen from the reassembly rules list (in sequence) so that different responses are provided to the user in the case where the same keyword happens to be matched in future. This initial lack of repetition gives the system the appearance of some form of intelligence, but when a reassembly rule is recycled (the reassembly rules list has been exhausted) then the "wizard behind the screen" has been exposed and the system's limitations revealed.

ELIZA has an inbuilt editor (or simply a reference to ED) which allows editing of conversation scripts on the fly. The rationale is that a conversational system can be built up over time with a small set of keyword and rules that is manually expanded with experience. Even though this does present a maintenance issue (with careful consideration of rule addition and/or modification requiring some degree of verification), the rationale is similar to that of the same rule maintenance issues addressed by RDR/MCRDR systems (Kang, 1995).

ELIZA's keywords are ranked, so that a higher-ranked keyword is chosen in preference amongst others in an input sentence. Multiple sentences in the one input are bounded by delimiters such as periods or commas, and each sentence is parsed separately. If a keyword is not found in an individual sentence, it is no longer considered and it is discarded, and if no keywords were found in the entire multi-sentence input, a default response is returned. This is specified in the rule script by the keyword "NONE", and the most general expression match decomposition rule condition, 0 (which is equivalent to a general wildcard match such as the more modern *). This response should be a context-free remark, designed to illicit more information or give the

illusion the system has considered the input. Examples include "*Please go on*", "*I see*", and "*That's very interesting*".

Simple decomposition rule matching consists of an early use of regular expression-like notation, with 0 matching multiple words, and any other numeral indicates an explicit number of words to match. For example, a decomposition rule such as:

```
(0 YOU 1 ME)
```

matches any number of words prior to the literal `YOU`, followed by one (1) word then the literal `ME`. So a sentence such as "*Everyone I know except you hates me*" would match this rule, but "*I know you do not like me*" does not match as the text between "*you*" and "*me*" is more than one word.

ELIZA does not learn from its experience and it can then easily be caught out in its actual lack of knowledge or the continuation of a dialog context. It discards each of its inputs, except for a subset of keyword matches (specified by a MEMORY tag), that can then be associated with a class e.g. "*Mother*" can match "*Family*".

2.3.1.2.2 PARRY

PARRY (Colby, 1971) was a system written in MLISP running on a PDP6/10 at Stanford that modelled non-human artificial paranoia, in that its responses to a psychiatric interview were judged by a clinician to be paranoid in behaviour, similar to real humans. Its model relies on calculated values for "affect variables" that elevate emotional or paranoid responses, such as fear and anger, and it adjusts the output depending on the threshold values of these variables. The threshold values can be arbitrarily set and adjusted to model slightly different behaviours. For example, if the fear level is elevated, responses might tend to ignore interviewer input and refer to other responses, which are indicative of perceived external, malevolent entities.

Keyword matching is used to determine the current topic, which then influences the responses to be more contextually relevant. Rules (known as conceptualisations) map input into two different structures, either a reference to an attribute of an object, or the relation of an object to another object.

Responses are also altered based on the current topic, which is set by scanning the input for keywords. Default responses are given if the input is not matched, with generic attributes given for the current topic.

2.3.1.3 Pattern Matching Scripting Languages

Two well-known and widely used general-purpose scripting languages for pattern-matching conversational agents are AIML and ChatScript.

2.3.1.3.1 AIML

The majority of Loebner Prize winners (see Section 2.3.1.4) have used or have been based on the XML-compliant language AIML (Artificial Intelligence Markup Language). This scripting language arose from the technology developed for the conversational agent, Artificial Linguistic Internet Computer Entity (ALICE) (Wallace, 2011, 2016).

Table 2.3: Example AIML (Marietto et al., 2013)

```
<aiml version="1.0.1" encoding="UTF-8"?>
<category>
        <pattern> HELLO BOT </pattern>
        <template>
                Hello my new friend!
        </template>
</category>
</aiml>
```

AIML models a stimulus-response paradigm. Input stimulus (text utterances) patterns are matched against AIML objects defined in scripts, in units of dialog or "knowledge-units" called categories (defined by an associated `category` XML tag), with the collection of all categories forming the knowledge-base (Marietto et al., 2013; Abdul-Kader and Woods, 2015). In addition to the `category` XML tag, there are two other main tags – `pattern` and `template`. `pattern` defines pattern-matching rules against user input, whereas `template` defines the AIML-based system's output. Table 2.3 shows a simple example of AIML code. Although this is a very simple example, AIML quickly becomes significantly complex and far beyond the capabilities of non-IT experts – the language specification defines at least 13 tag types, with provisions for conditional processing, randomisation and complex pattern-matching syntax.

In addition to AIML-based conversational agents that are developed purely for the kudos in prize-attainment exercises like the Loebner prize, some application examples of AIML use include:

- virtual assistant-based tutoring in e-learning platforms such as T-BOT, Q-BOT and CHARLIE, (Mikic et al., 2008, 2009; Fonte et al., 2009);

- an *assistant tutor* in a multi-agent system that provides answers to students and other tutors through monitoring of a discussion forum (Alencar and Netto, 2011);
- an evaluation platform to assess qualitative and quantitative measures such as "usefulness" and "localizability" of various categories of conversational agent domains (AbuShawar and Atwell, 2016);
- a support system to act as an interface for the general public to query governmental systems such as land record maintenance and other GIS systems (Mahapatra et al., 2012);
- Chinese-language systems, such as student coursework counselling and question-and-answering for library referencing help by a talking robot (Fei et al., 2011). Wei et al. (2016) detail AIML's deficiencies in word segmentation for Chinese script due to its Western language bias, for example, word order in Chinese is less rigid compared to English.

The example studies above evaluate AIML-developed systems against varied testing data corpora and other quantitative and qualitative evaluation criteria typically by measuring the end-system stimulus to response in some manner (AbuShawar and Atwell, 2016), but they do not consider the degree of difficulty or skillsets required by the conversational author to create the conversational knowledge in the first place – it is clear that AIML is a *scripting language*, and development and maintenance of the knowledge corpus for any domain requires the assistance of software professionals to write the scripts (*AIML Tutorial*, 2020; Wei et al., 2016; O'Shea, 2014).

2.3.1.3.2 ChatScript

ChatScript (Wilcox, 2011) was developed to overcome perceived limitations in AIML.

Wilcox, in his justification for the creation of ChatScript states:

> "*AIML is a miserable system for authoring. As a programmer, I think of AIML as recursive self-modifying code at the assembly level of language processing. In other words, a content creation and maintenance nightmare. And its pattern matching is feeble.*"

In their review of Loebner Prize winners Bradeško and Mladenić (2012) offer the opinion that ChatScript is the successor to AIML. The rationale is that AIML suffers from its limit of concise expressiveness. Wildcards present in a rule's (category) condition explicitly match 1 or more terms. Additional rules are required for those patterns in the input needing a zero-term

match for the same condition. As rules explicitly pattern match specific words, this makes the treatment of synonyms extremely difficult – multiple rules are subsequently required to handle similar (synonymous) patterns that could be considered semantically the same. ChatScript offers the concept, which is simply a list of synonyms that can match the current rule's pattern, in addition to other programming features such as variables and functions. An ontological approach (WordNet (Miller, 1995)) is included for synonym definition, as well as integrated part-of-speech (POS) tagging. An example script is shown in Table 2.4 that shows defining the concept *meat* and its associated synonyms. The pattern-matching expression "I love *meat*" if satisfied, would then result in the text "*Do you really? I am a vegan*" as a response. Like its predecessor AIML, ChatScript also requires considerable software programming expertise to author its scripts.

Table 2.4: Example ChatScript (Bradeško and Mladenić, 2012)

concept: ~meat (bacon ham beef meat flesh veal lamb chicken pork steak cow pig)
s: (I love meat) Do you really? I am a vegan.

2.3.1.3.3 Semantic-Based Conversational Agents

Semantic-based conversational agents (SCA) are proposed by O'Shea (O'Shea, 2014). These agents would use a measure based on semantic sentence similarity, reportedly to overcome the issue of variations in skill of script-writing ability of different authors. Traditional conversational agents typically look at the structural components of sentences and pattern match, which the author claims make the maintenance task of producing multiple variations a high-overhead task for the script author – they must anticipate different ways users can articulate the same question. Using the SCA approach, scripts are composed of NL sentences. A sentence similarity measure is then applied against input sentences to match the pre-specified script content. A scripted sentence with the highest rank of similarity is chosen to be fired, producing the required response. An issue then is determining pre-determined thresholds in similarity matching – if the threshold is too high, no sentences will match and consequently an appropriate output will be missed. If the threshold is too low, inappropriate sentences will be matched that should not have been.

The sentence similarity measure is determined using a statistical approach, "*Sentence Similarity based on Semantic Nets and Corpus Statistics*". This approach considers the location, word order and frequency of words in different contexts using a combination of two measures: word similarity – based on a calculation derived from an external hierarchical database, WordNet (Miller, 1995) of synonyms and word order; each word in the combined comparison sentences (with

redundant repetitious words removed) is indexed and a simple calculation is made comparing word order indices of the original sentences with the combined one. The final sentence similarity is derived from both measures, with the word order measure weighted to decrease its significance in the final calculation.

The user input's similarity measure is calculated against all rules (natural language sentences) in the current domain script.

O'Shea asserts that editing SCA scripts are a low maintenance task. However, the script writer is still required to learn and use the necessary script syntax and an added complexity is that threshold similarity measures need to be established. An equivalent number of rule-nodes are still needed compared to other scripted pattern-matching implementations, but the complexity of considering alternate pattern rules for the same equivalent semantic meaning is reduced.

2.3.1.4 Conversational Agent Development Encouragement

As a subjective measure of the success of a particular CA implementation and approach, frequent references are made in the literature and popular press to the *Loebner Prize* (Majid al Rifaie, 2018; Mauldin, 1994; O'Shea, 2014). Established in 1991, this is an evaluation of the Turing Test (Turing, 1950) named after the British Mathematician, Alan Turing. The prize is awarded annually to the conversational agent that is judged the most human-like (relative to other entries in the same year). Evaluation is conducted in four rounds; in each round, each judge (of four) interacts with two entities via a terminal - one will be an actual human, the other an AI conversational agent. If the conversational agent can convince half of the judges that it is the human, then the conversational agent's creator will be awarded a Silver medal and a monetary prize ($25,000 USD in 2015). If there is no clear winner under these circumstances, then smaller prizes are given per the judge's ranking (ranging from $4000 down to $500) for first to fourth place.

The results (and subjective scoring system) of this competition do not appear to be a basis for a rigorous, scholarly and peer-reviewed evaluation of the systems scrutinised, rather it is a driving, populist approach used to encourage the development of such systems and foster public interest in AI in general. The competition has been heavily criticised as a *controversial publicity stunt* according to established researchers such as Marvin Minsky (Sundman, 2003).

Mitsuku (Worswick, 2019) is the most-recent and most frequent winning conversational agent (2013, 2016 – 2019) based on AIML and the ALICE conversational agent (Wallace, 2016). It can respond to domain-specific queries using targeted AIML script files.

2.3.2 Natural Language Interfaces to Databases (NLIDB)

One aspect of this thesis is to define a method for an easily-maintainable conversational agent system that has a natural language interface. The resulting augmentations to a KBS methodology like MCRDR include database interaction via predefined queries executed as consequences of satisfied rules. This interaction will then in effect also produce an NL interface (via the pattern-matching and contextual augmentations performed) to databases, so it is worth investigating the literature relevant to NL interaction with databases. This field is known as NLIDB.

NLIDB systems generally convert natural language input questions into a specific database query language for the domain in which they are used (Grosz et al., 1987; Li et al., 2005; Bais et al., 2016; Nguyen et al., 2016, 2017), or domain-neutral ontologies (Chaw, 2009). An advantage of NLIDB is that non-computer-specialists are not required to learn and use formal query languages to interact with database management systems, but the practicality is that the natural language input is usually constrained to a limited subset of the language (Androutsopoulos et al., 1995). This can lead to user frustration as the actual linguistic coverage by the system is not necessarily obvious – the user must be trained to a degree as to what types of questions the system is capable of understanding, which is the brittleness problem inherent in knowledge-base systems (Section 2.2.2.3). Such ambiguity is not present when a formal query language is used.

2.3.2.1 NLIDB and Brittleness

To address the definition of brittleness (users are unfamiliar with the content and structure of a knowledge-base (Chaw, 2009)), the conversational agent aspect of this thesis is considered, and, for example, its applicability as a natural language interface to databases. One of the main advantages for a user in such systems is that they are not required to learn and adopt technical database query formalisms (Smith et al., 2014), but the practicalities of the system's linguistic constraints may quickly lead to frustration – the user must overcome the inherent brittleness of the system and still be conversant in the linguistic coverage as to what and cannot be understood (Androutsopoulos et al., 1995), while unfortunately, at the same time, they must possess a mild appreciation of underlying database schemas. Ontological approaches (Lenat et al., 1985; Miller, 1995; Li et al., 2005) try and mitigate brittleness through the extension of mapping terms to *concepts* or "common-sense" and semantic meaning by massive datasets such as OpenCyc (*OpenCyc*, 2018), Cyc (*The Cyc Platform*, 2020) or WordNet (Miller, 1995), or alternatively, for example, using machine learning techniques to classify concepts in Wikipedia (Gabrilovich and Markovitch, 2006), instead of direct, literal matching. A smaller-scale localised ontological-

style approach is by paraphrase generation which automatically generate synonyms which are in effect, semantically equivalent terms or sentences (Oh et al., 2015; Madnani and Dorr, 2010; Huang et al., 2019; Yang et al., 2019; Hunt et al., 2019). The paraphrasal method is adopted in this doctoral study, although the approach taken was that the lexical or phrasal paraphrases are directly specified by the domain expert when populating the system's associated *dictionary* – future work will investigate the automatic extension and generation of paraphrasal terms.

2.3.2.2 Syntactic grammar systems

In syntactic grammar systems, input questions are first parsed syntactically to a parse tree that is directly matched to a database query language expression.

LUNAR (Woods et al., 1972) is the best-known early example of a syntactic grammar NLIDB. Implemented in LISP on a PDP-10, the system allowed scientists to query a chemical analysis database of lunar samples using English.

For example, the simplified grammar shown in Table 2.5 (Androutsopoulos et al., 1995) is similar to the form used in LUNAR. The syntax specifies a sentence (S) consists of a noun-phrase (NP) followed by a verb-phrase (VP). An NP is a determinant (Det) such as "what" or "which" followed by a noun (N). Nouns are specified explicitly, as are verbs. The combination of the domain specificity of the syntax rules, application-specific nature of the database, coupled with difficulty in determining rules to map the parsed tree into query expressions (Nguyen et al., 2017), meant these system are not easily ported to other domains.

Syntactic trees are still used for modern NLP techniques, for example, Sidorov et al. (2014) parse syntactic trees to construct their so-called *syntactic n-grams* (sn-grams) which are n-grams (see Section 2.3.3.2) where the n-gram terms are neighbouring syntactic relations as opposed to characters or words in the source lexicon. They then use sn-grams as features for machine learning where n-grams are traditionally used.

Table 2.5: Simplified LUNAR input parse grammar syntax

Term		**Expansion**
S	→	NP VP
NP	→	Det N
Det	→	"what" | "which"
N	→	"rock" | "specimen" | "magnesium" | "radiation" | "light"
VP	→	V N
V	→	"contains" | "emits"

2.3.2.3 Semantic grammar systems

Semantic grammar systems also parse and map input questions to a parse tree, but the non-leaf nodes in the parse tree (the grammar categories) do not necessarily directly correspond to syntactic concepts (Androutsopoulos et al., 1995). Androutsopoulos et al. (1995) give an example of a semantic grammar (Table 2.6) and an example parse tree (Figure 2.7) for the input sentence, "*which rock contains magnesium?*". The example shows, for instance, the term `Substance`, does not match a *syntactic* component such as a noun, noun-phrase etcetera.

Table 2.6: Example semantic grammar syntax

Term		**Expansion**
S	→	Specimen_question | Spacecraft_question
Specimen_question	→	Specimen Emits_info | Specimen Contains_info
Specimen	→	"which rock" | "which specimen"
Emits_info	→	"emits" Radiation
Radiation	→	"radiation" | "light"
Contains_info	→	"contains" Substance
Substance	→	"magnesium" | "calcium"
Spacecraft_question	→	Spacecraft Depart_info | Spacecraft Arrive_info
Spacecraft	→	"which vessel" | "which spacecraft"
Depart_info	→	"was launched on" Date | "departed on" Date
Arrive_info	→	"returns on" Date | "arrives on" Date

An issue with semantic grammar systems, similar to syntactic grammars, is that the grammar is very domain-specific as a new grammar has to be written if the system is to be ported to another domain. The generation of these grammar rules has to be facilitated in a natural, easy to maintain manner.

LADDER (Hendrix et al., 1978), implemented in LISP, facilitated natural language querying using semantic grammars to a distributed database consisting of remote networked DBMS in the Naval domain. An intermediate query form is parsed against templates for heterogeneous end-system query languages.

PLANES (Waltz, 1978) was developed at the University of Illinois, and it included an English front-end to allow users to query the system about aircraft flight and maintenance data. The system provided limited clarifying discourse to allow question refinement and/or resolution of ambiguity. Parsing was accomplished by mapping input against phrase patterns stored as augmented transition networks (ATNs) – first proposed by Woods and Bobrow (Woods and Bobrow, 1970), which allowed the recognition of phrases with specific meaning, and semantic

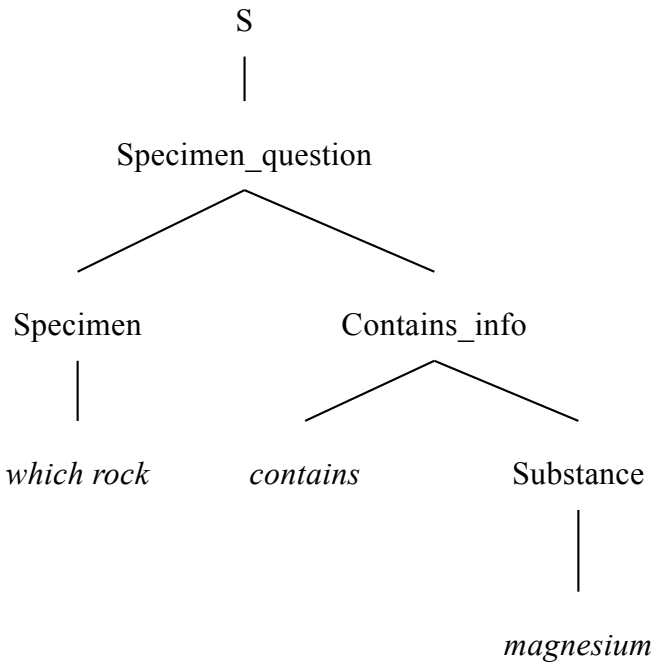


Figure 2.7: Example semantic parse tree

sentence patterns termed concept case frames. The results of matching (an intermediate form) are then translated into a specific formal query language.

CHAT-80 (Warren and Pereira, 1982) was developed entirely in Prolog and evaluated on a DEC PDP-10. English question phrases were transformed directly into Prolog expressions, which, after some optimisation, were then executed against the Prolog rule base. A geography domain was chosen for testing, allowing constrained natural language querying against a geographical database containing facts about features such as oceans, seas, rivers, cities and countries. The system contained a small set of English vocabulary words – enough to sufficiently query the underlying database encoded in Prolog.

2.3.2.4 More recent NLIDB systems

PRECISE (Popescu et al., 2004) uses machine learning (statistical parsers), in particular, the Charniak parser (Charniak, 2000) to produce a syntactic parse based on part-of-speech (POS) tags, but they initially re-train the tagger model with an additional set of manually-labelled POS tags for 150 questions presumably related to the target domain. The resulting parse tree more accurately reflects the target domain, but they then correct syntactic errors by using *semantic overrides* based on a lexicon produced automatically from the target database schema

(for example, attribute (column) and relation (table) names). On their evaluation dataset, PRECISE achieved an accuracy of 94% (compared to ≈ 62% using the generalised tagger model). Popescu et al. (2004) claim PRECISE is database independent, however it requires retraining for each domain, as well as semantic analysis of each database schema.

Intermediate XML Logical Query (IXLQ) (Bais et al., 2016) is an intermediate representational approach that takes natural language queries (NLQ), performs a syntactic analysis (after morphological tokenisation) to produce a syntactic parse tree. They then perform a semantic analysis mapping syntactic rules to associated semantic rules to produce the intermediate, logical XML query description. The authors note the generated logical table and attribute names are tested against existing database schemas for matches, and, if lacking, domain-specific schemas are then used. Their system correctly generated ≈ 89% of queries from 13493 test NLQ inputs and the majority of errors stem from the semantic analysis state. It should be noted the queries are relatively simple – selection with some aggregation (count, max, min) but no table joins appear to be generated.

Natural Language Interface to XML (NaLIX) (Li et al., 2005), in contrast to IXLQ, supports value joins. An English sentence (NLQ) is ultimately parsed to an XQuery expression. The parsing process classifies known expressions as tokens (that are directly mapped to XQuery components), and others as *markers* which will be validated though an iterative, interaction session with the user to help define semantic relationships with known tokens. Ontological or paraphrasal matching techniques based on WordNet (Miller, 1995) to assist with database schema matching.

The ASKME system (Chaw, 2009) uses a specially restricted version of English called Computer Processable Language (CPL) (Clark et al., 2005) to avoid issues such as ambiguity that arise in NL processing of general English. CPL – which has restrictions on grammar, style and vocabulary, is interpreted by a syntactic parser, however it still requires a degree of user training to teach them how to construct their requests.

Natural Language Interface for Relational databases (NaLIR) (Li and Jagadish, 2014*b*,*a*) uses an "off the shelf" NL parser (the Stanford Natural Language Parser (De Marneffe et al., 2006)) to produce a semantic parse tree. Like NaLIX, the user is interactively asked to refine phrases that are not recognised as matching database schema items, and refinement includes ranking of adjusted parse trees that match existing semantic coverage (i.e. that are understood by the system in that they explicitly correspond to an SQL statement) to produce what the authors term a *query tree*. Unlike NaLIX, which repeatedly requires the user to rephrase a query until it understood, NaLIR also automatically rephrases the query for user acceptance. The final

result is parsed by a mapping of parse-tree tokens to SQL code (based on a knowledge-base of enumerated phrases), final schema mapping is achieved through a similarity measure (when multiple candidates exist, the user is prompted to choose), and due to the interactive refinement and user verification, this almost always results in correctly translated SQL.

The SEEKER and Aneesah systems (Smith et al., 2014; Shabaz et al., 2015) are similar NLIDBs coupled to CAs. SEEKER uses a commercial CA to map appropriate SQL templates to evaluate against an underlying database and in a similar manner to NaLIX and NaLIR, it allows a targeted followup of refinement of query results via a GUI. CA scripts are used to pattern-match keywords in utterances, and then the most appropriate SQL template is chosen via an expert system based on matched variables in the utterance; the SQL templates are produced from a pre-establishment phase requiring a lengthy questionnaire capturing common NL phrases used and their resulting associated queries (Smith et al., 2014). In contrast, although somewhat similar, Aneesah (Shabaz et al., 2015) dynamically builds an SQL query (although it is not clear how the mapping of utterance to query is performed) after a CA engine utilises a scripted-capture of user utterances. In both cases misinterpretations require careful, offline maintenance of the CA engine scripts and both are evaluated via satisfaction and task-based questionnaires of very small participant groups (10 and 20 participants respectively). A key component of SEEKER is the ability to further refine query results.

An alternative approach, instead of pre-defining query templates to potentially map to user input, is to automatically generate all search strings based on a corpus of existing questions in community question answering (cQA) platforms such as *Stack Exchange* (Figueroa, 2017). This means a very large corpus of queries is generated against the existing knowledge-base that can then be used to provide related questions (and answers) to a user's initial query. The authors extract a number of attributes from a query based on computed results from various sources including CoreNLP (Manning et al., 2014) and WordNet (Miller, 1995) – such as sentences (using part of speech tagging, named entity recognition and sentiment analysis), semantic connections, and others.

In this thesis research a natural development or outcome of the implemented C-MCRDR CA system is a form of NLIDB, however the database query generation stage from a natural language query (NLQ) is not conducted using statistical (for example, POS tagging) parsing of the NLQ. The initial work during knowledge acquisition requires the domain expert to manually create queries (in a guided, abstract form, expressed ultimately as an XML intermediate form) that are then pattern matched to the NLQ as a result of C-MCRDR inference. Although the simplicity of the approach addresses the research questions defined in Section 1.2.1, a POS

tagger approach (Manning et al., 2014) could easily be adopted for tokenisation labelling and subsequent rule attribute specification during rule maintenance. This however would require significant cognitive and grammatical expertise by the domain expert to create conversational rules, but it is a possible avenue of future research.

2.3.3 Approximate String Matching

As previously mentioned, the most widely-used technique for conversational agents to process natural language inputs is pattern matching (Bradeško and Mladenić, 2012). When considering *how* natural language text is pattern matched, there are several possibilities that are categorised as approximate string matching techniques. Approximate string matching techniques are used when comparing a required string with other strings that are similar (but not necessarily identical). There are many string matching techniques, categorised by function and type that have resulted from several decades of research and algorithm development (Hakak et al., 2019), however, such techniques are not directly employed in this thesis (although the string edit distance (Levenshtein, 1966) is used indirectly in evaluation via the definition of word error rate, WER – please see Section 2.3.4.1, Equation 2.8). Instead, key term matching, via a lexical paraphrasal (Fader et al., 2013; Oh et al., 2015; Li et al., 2005) or ontological (Miller, 1995) approach (by *exact* string matching or rule-based regular expression matching) is used (see Section 3.3.2).

Hakak et al. (2019) classifies the direct or exact string matching technique primarily used in this thesis (apart from the rule-based regular expression generalisations) as the *Character Comparison Approach*, although other algorithms such as the *Hashing Approach* (computing hashes of individual strings for numerical comparison) could perhaps have been adopted. When the source utterance string to be used for inference is received as a result of ASR transcription, the typical possibility exists of phonetically-similar strings against terms in the paraphrasal list (i.e. homophones) being introduced instead of the original source intent. Such terms are preprocessed to correct common errors (see Sections 3.3.9.1 and 4.2.9) but an alternative approach not taken could have adopted approximate (Hall and Dowling, 1980) or phonetic string matching techniques for correction. Approximate string matching is especially useful when translating out-of-vocabulary (OOV) terms such as names and technical terms (Järvelin et al., 2016) and is typically used in text retrieval systems (Navarro, 2001).

A brief overview of typical techniques (that can be applied at the individual word or *unigram* matching) thus follows, with matching methods that range from using simple similarity metrics (measuring the *distance* between two strings in a numerical manner) to other methods that

use language-specific phonetic representations of words. Sentence matching techniques, such as cosine similarity, term frequency or inverse document frequency (TF-IDF) are more appropriate measures for determining document similarity in, for example, document retrieval or classification problems and so they are not considered here.

2.3.3.1 String Edit Distance

The string edit distance (ED) (Levenshtein, 1966; Hall and Dowling, 1980; Navarro, 2001; French et al., 1997; Zobel and Dart, 1995) is the most widely known string metric (Tissot and Dobson, 2019; Navarro, 2001) although applications range from DNA analysis to studies of bird song (Kruskal, 1983). String edit distance operates between two input strings s and t, with lengths $|s|$ and $|t|$ respectively, via a recursive definition, $ED(|s|, |t|)$, given below (modified from the form given by Zobel and Dart). The function returns the minimum number of single-character edits (substitutions and deletions) needed to transform the string s into t.

$$
\begin{aligned}
&ED(0,0) = 0 \\
&ED(i,0) = i \\
&ED(0,j) = 0 \\
&ED(i,j) = min\big\{ED(i-1,j)+1, ED(i,j-1)+1, ED(i-1,j-1)+d(s_i,t_j)\big\} \\
&\quad \text{Given } s_i \text{ is the } i^{th} \text{ character of } s, \text{ and} \\
&\qquad\quad t_j \text{ is the } j^{th} \text{ character of } t, \text{ and} \\
&d(a,b) = \begin{cases} 0, \text{if } a = b \\ 1, \text{if } a \neq b \end{cases}
\end{aligned}
$$

The string edit distance is an efficient and effective method for ASR transcription error detection/correction (Fiscus et al., 2006; Raghavan and Allan, 2005; Bisani and Ney, 2004; Järvelin et al., 2016; Zobel and Dart, 1995) as it in essence is used in part as the word error rate metric for error detection, but also as a numeric threshold when considering plausible terms for correction (at the individual character level for isolated words). Raghavan and Allan (2005) found when matching name variants (when considering named entity recognition, NER) from the output of different ASR systems (which in effect are OOV terms) compared to the lexicon of each ASR system, that edit distance was an efficient (and better performing) technique. A similar premise applies when the source string is a result of optical character recognition (OCR) (Järvelin et al., 2016).

As previously mentioned, a string edit distance or n-gram variant may be considered in future work when *correcting* ASR transcription errors at the individual word level, in contrast to its use in error detection.

2.3.3.1.1 Hamming Distance

The Hamming Distance (Hamming, 1950; Kruskal, 1983; Navarro, 2001) is a variant of edit distance and it is defined as the minimum number of substitutions that would be needed to transform one string to the other. In other words, it is a measure of the number of characters that differ between two strings. Hamming Distance however is limited to equal length strings.

2.3.3.1.2 Longest Common Subsequence

Longest Common Subsequence (Needleman and Wunsch, 1970; Kondrak, 2005; Cormen et al., 2009) (LCS) is simply the length of the maximum-length common substring between two strings. As a measure this method suffers from the fact it is biased by the length of the strings being compared, so an alternative (and preferred) method is to normalise the result by dividing by the length of the longest string, producing the longest common subsequence ratio (LCSR) (Kondrak, 2005). Although LCS was found to be the better method for word matching (in comparison to bigrams and trigrams, see Section 2.3.3.2), it has a much higher computational overhead in comparison, and is considered inferior to string edit distance and s-grams (Järvelin et al., 2016).

2.3.3.2 n-grams

An n-gram (or *s-gram, q-gram*) is a widely used technique (Järvelin et al., 2007; Keskustalo et al., 2003; Sidorov et al., 2014), and in this context for approximate string matching is defined as a contiguous substring (of length n) of a parent string (of length l) (Ukkonen, 1992; Cavnar et al., 1994; Shannon, 1948; Järvelin et al., 2007). The more recent term of s-gram refers to n-grams that can be composed of non-adjacent characters and the choice of n influences the effect or performance of word matching – trigrams ($n = 3$) and bigrams ($n = 2$) achieve the best results (Pfeifer et al., 1996; Järvelin et al., 2007).

2.3.3.2.1 n-gram count

The simplest similarity measure between two strings when using n-grams is to choose a value of the n-gram size (n) and then determine how many n-grams both strings have common – see

Equation 2.1 (Zobel and Dart, 1995; Kondrak, 2005).

$$ngcount_n(s,t) = |G_s \cap G_t| \tag{2.1}$$

where s and t are strings,

G_s and G_t are the sets of n-grams in s and t respectively

This measure however does not take into account any notion of the position of each n-gram in the source string, for example, $ngcount_2(\text{“}abcdef'', \text{“}efcdab'')$ and $ngcount_2(\text{“}abcdef'', \text{“}efdebc'')$ both give the same measure (3), even though in both cases distinctly different strings are being compared.

2.3.3.2.2 n-gram distance

Ukkonen (1992) defined the n-gram (or *q-gram*) distance to allow for differences in string lengths (Equation 2.2, (Ukkonen, 1992)).

$$ngdist_n(s,t) = \sum_{g \in G_s \cup G_t} |N(s,g) \cap N(t,g)| \tag{2.2}$$

where s and t are strings,

G_s and G_t are the sets of n-grams in s and t respectively

$N(x,g)$ is the number of occurrences of n-gram g in string x

Zobel and Dart found n-gram distance approximately equal in performance to string edit distance (in name matching) but it was better performing when considering spelling correction evaluation (Zobel and Dart, 1995). Conversely, around the same time, Pfeifer et al. (1996) found n-grams better than string edit distance.

2.3.3.2.3 Sørensen-Dice Coefficient

Another common similarity measure using n-grams for string matching is the Sørensen-Dice coefficient (Equation 2.3 (Sorensen, 1948; Dice, 1945)) which is also the F1 score when rearranged (Equation 2.4). The use of this similarity measure in effect, finds the nearest neighbour match in a set of strings (the dictionary) for candidate correction (Angell et al., 1983), and it is commonly used with bigrams (Kondrak, 2005). The bigram approach in particular is criticised as it often cannot detect a similarity between strings that have no n-grams in common, but that are otherwise very similar, and conversely the measure can give a high similarity (for example, 1) for strings that have the same sets of n-grams, but the n-grams actually occur in different positions (Kondrak, 2005).

$$ngdice_n(s,t) = \frac{2|G_s \cap G_t|}{|G_s| + |G_t|} \tag{2.3}$$

$$ngF1_n(s,t) = \frac{2}{\frac{|G_s|}{|G_s \cap G_t|} + \frac{|G_t|}{|G_s \cap G_t|}} \tag{2.4}$$

where s and t are strings,

G_s and G_t are the sets of n-grams in s and t respectively

2.3.3.2.4 Jaccard Distance

The Jaccard Distance is defined in Equation 2.5, which is 1 subtract the *Jaccard similarity* (Pfeifer et al., 1996). Although a popular measure, the "*Jaccard Distance is insensitive to the counts of each n-gram in the strings to be compared*" (Järvelin et al., 2007).

$$ngJaccard_n(s,t) = 1 - \frac{|G_s \cap G_t|}{|G_s \cup G_t|} \tag{2.5}$$

where s and t are strings,

G_s and G_t are the sets of n-grams in s and t respectively (2.6)

2.3.3.3 Soundex

Soundex, and variations of the idea, employ phonetic matching. Zobel and Dart state "*Phonetic matching is used to identify strings that may be of similar pronunciation, regardless of their actual spelling*" (Zobel and Dart, 1996). Soundex (Hall and Dowling, 1980), was developed by Odell and Russell and patented in 1918 by Russel (Mokotoff, 1997) and it refers to codes that are based on letter sounds, reducing a word to a maximum of four characters. Table 2.7 shows the Soundex letter-mapping-to-code rules along with the algorithm itself (the version shown is the more recent American Soundex System). Soundex is primarily used as an indexing function for surnames, but unfortunately, unlike a hash function with low or non-existent hash collisions, dissimilar-sounding words can be transformed to the same Soundex code, whereas words that are close phonetically can produced entirely different Soundex codes. Soundex is also based on English pronunciation so this can be limiting when considering other languages. Zobel and Dart (1995) found edit-distance techniques to be more effective (in terms of matching performance) compared to phonetic techniques.

Table 2.7: Soundex codes and algorithm (Zobel and Dart, 1996; Mokotoff, 1997)

Code	**0**	**1**	**2**	**3**	**4**	**5**	**6**
Letters	aeiouyhw	bpfv	cgjkqsxz	dt	l	mn	r

Algorithm

1. Replace all but the first letter of the string by its phonetic code.
2. Eliminate any adjacent repetitions of codes.
3. Eliminate all occurrences of code 0 (that is, eliminate vowels).
4. Return the first four characters of the resulting string.

2.3.3.4 Phonix

Phonix (Gadd, 1988, 1990) is a derivation of Soundex (but with two more end-mapping codes), primarily used for online public access catalogs (OPAC), where a transformation to groups of letters is first applied before the letter-to-code mapping. Phonix has 163 transformations (Bakar et al., 2000; Zobel and Dart, 1995). These consist of letter combinations (for example gn, ghn and gne map to n), which allow for additional phonetic context that Soundex does not consider, such as distinguishing between c and s. The codes generated are similar to Soundex (see Table 2.8), except words beginning with vowels have the vowels replaced by 'v' (Bakar et al., 2000), and the result can be up to 8 individual code digits. Interestingly, the evaluation performed by Zobel and Dart (1995) found Phonix performed worse than Soundex and that phonetic string matching in general performed poorly:

> "*…we found Phonix to be worse than Soundex, despite the fact that it is some seventy years more recent and is explicitly designed as a Soundex replacement.*
> …
> *The most dramatic aspect of our results is that they demonstrate that phonetic coding is a poor string matching mechanism. It can provide some speed and space advantages for coarse search, but at low effectiveness…* "

The recent work of Tissot and Dobson (2019) shows phonetic approaches to string matching are still relevant. Their research uses a hybrid approach, coupling string and phonetic measures to match misspelt names of drugs from written medical records (written in Portuguese).

Table 2.8: Phonix codes (Gadd, 1990)

Code	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Letters	aehiouwy	bp	cgjkq	dt	l	mn	r	fv	sxz

2.3.3.5 Other Well-Known Phonetic Variations

Other variations include Metaphone (Philips, 1990) which segments English words into phonetic terms, and it was developed to address issues with Soundex. It is more sensitive to positional variation of letters, and letter combinations such as TH (Hood, 2002).

Caverphone (Hood, 2002, 2004) is similar to Metaphone, but it "*…was designed to be similar to Metaphone in the sensitivity to letter placement, but with an encoding scheme intended to allow for the range of accents present in the study area (southern part of the city of Dunedin, New Zealand) in the years 1893-1938*". The second version of the algorithm was generalised to allow for arbitrary English phonetic matching (Hood, 2004).

Double Metaphone (Philips, 2000*a*,*b*) can give two encodings for a string to allow for shared ancestry when matching surnames and/or words that can be pronounced in more than one way, and it corrects deficiencies and errors introduced in Metaphone while allowing for more alternate pronunciations in other non-English languages such as the Romantic languages like Italian and French, to Germanic and Slavic languages. A recent example of Double Metaphone in research is a context-sensitive spelling correction system for medical records (Fivez et al., 2017).

Performance evaluation of the phonetic variation methods (including Soundex) found there was no best technique in general (Koneru et al., 2016). Metaphone was best for English dictionary words, Soundex had better accuracy in spelling correction, but lower precision.

2.3.4 Intelligent Personal Assistants (IPA)

The final considerations needing discussion here that are associated with the overall aims of this doctoral study relate to the measurement and correction of automatic speech recognition errors of Intelligent Personal Assistants. Intelligent Personal Assistants are coupled speaker-and-microphone devices designed for mass-market use. With the ever-growing market of "smart" devices and the Internet of Things (IoT), the list of capabilities IPA devices gain to control and assist with aspects of modern life is considerable (Lopatovska et al., 2018). Evaluation typically compares psychological issues, such as cognitive load (e.g. how easy are the devices to use) (Strayer et al., 2017), social engagement (elderly sociability) (Reis et al., 2017; Oh et al., 2020), user satisfaction and variation in accents (Jiang et al., 2015; Pal et al., 2019), privacy concerns (Manikonda et al., 2017; Horwitz, 2018; Hoy, 2018), learning languages (Dizon, 2017; Dizon and Tang, 2019), and a comparison of services by survey (Hoy, 2018; de Barcelos Silva et al., 2020), including considerable usage in health care (Montenegro et al., 2019). These

studies typically do not evaluate the actual ASR transcription error rates (measured by word error rate, (WER)), possibly due to the device's performance achieving a level where they are not usually perceptible to the end users, instead the surveys are generally qualitative in nature, such as the number of skills supported, user satisfaction and deployment scenarios, and focusing on the device's associated conversational agent.

2.3.4.1 ASR Error Detection and Correction

ASR systems are typically evaluated by analysing the WER (Park et al., 2008; He et al., 2011; Bisani and Ney, 2004; Raghavan and Allan, 2005; Fiscus et al., 2006). These errors typically occur either as substitutions (S), insertions (I) and deletions (D) (Zhou et al., 2005) *at the word level* (and not individual character level) that occur for a given source of words, N. This leads to the definition of the Word Error Rate (WER, Equation 2.7) :

$$WER = \frac{S + D + I}{N} \tag{2.7}$$

An alternative notation is given below (Bisani and Ney, 2004), where WER is explicitly defined in terms of the edit distance (Levenshtein, 1966), e_i:

$$WER = \frac{\sum_{i=1}^{N} e_i}{\sum_{i=1}^{N} n_i} \tag{2.8}$$

n_i is the number of words in sentence i,

e_i is the *edit distance* between the ASR output (at the word level)
and the reference words of sentence i,

N is the total number of sentences

Microsoft researchers argue that WER is not the best metric for speech recogniser *training* as it "*takes no consideration of the contextual and syntactic roles of a word*" (He et al., 2011). In particular their research is considering contextual information for language translation of English to Chinese using different metrics such as measuring n-gram (see Section 2.3.3.2) matches between the translation and the reference (Bi-Lingual Evaluation Understudy, BLEU), which uses a weighted average of variable length phrase matches against the reference translations (Papineni et al., 2002), which could be considered a variation of the NIST score method (Doddington, 2002), and Translation Edit Rate (Snover et al., 2006), a measure of the number of edits needed to fix the translation output so that it semantically matches a correct translation. Note how-

ever, the research and such metrics are used for translation and training, not a measure of non-contextual error.

Schemes using confidence scores and intermediate probabilistic matches (Chen et al., 2013; Pellegrini and Trancoso, 2010; Zhou et al., 2005) to detect transcription errors require that information to be available i.e. inside the speech-to-text pipeline. Without access to the internal processes (a blackbox approach) this is not possible. Error detection typically requires features of the ASR system such as confidence scores and hypothetical (possible match) word lists to be available. Zhou et al. (2005) use data mining techniques, such as Naive Bayes, Neural Networks and SVMs to classify errors, and they apply these to large domain vocabularies that feature continuous speech, termed Dictation Speech Recognition (DSR). This is in contrast to the smaller vocabulary needed in the evaluation domain (see Section 4.2), and the transcription results utilising the existing *Google Actions* and *Alexa Skills Kit* which take a "blackbox" approach do not include confidence scores. This limits the error detection to comparisons with original source text, or other statistical means, such as testing for out-of-vocabulary (OOV) words. Other error detection approaches include classifiers using features not generated by the ASR decoder, such as word-boundary detection by signal-level acoustic-prosodic features, confusion network density, and linguistic features utilising for example, part of speech (POS) tagging and Names Entity Detection (NED) (Chen et al., 2013).

Other schemes process signal-level acoustics and filtering for both detection and improvement (Moore et al., 2017; Li et al., 2017), but these are acoustical and electrical engineering approaches and are beyond the scope of this doctoral study – transcription results from the hardware are limited to "as-is" and consequently the approach is taken to accept that errors occur and to manually measure the rates under standard, repeatable conditions. Recognition rates will change based on speaker vocalisation variability (for a single speaker and across different speakers) and spoken corpus (O'Shaughnessy, 2008). Distinguishing between speakers and the inherent 'open-channel' security implications (Hoy, 2018; Horwitz, 2018)) are an active area of research, such as the VAuth speaker authentication system (Feng et al., 2017). No considerations are undertaken to measure and distinguish errors that occur due to pronunciation variabilities between different speakers (Jeanrenaud et al., 1995), although this is an important consideration for future research.

Generally ASR transcription *correction* is conducted through a manual process, selection between alternative probabilistic (most-likely) hypotheses, post-transcription editing (Ringger and Allen, 1996) or pattern-based learning (Mangu and Padmanabhan, 2001). Unsupervised methods for correction basically take statistical approaches (Bassil and Semaan, 2012; Sarma and

Palmer, 2004), for example, looking at phonetic similarity with region-based subsets of the target vocabulary, this includes various string matching techniques like Soundex (Hall and Dowling, 1980), Phonix (Gadd, 1988), string edit distance (Hall and Dowling, 1980; French et al., 1997) and Editex (Zobel and Dart, 1996). ROVER (Fiscus, 1997), like the probabilistic alternative selection approach, used a voting mechanism to choose the best composite output based on the input of several ASR systems to reduce the word error rate. A similar approach is taken by an ensemble-based error correction system that uses multiple language models to collaboratively arrive at a decision for the "best" sentence after threshold error level is detected (Ikegami et al., 2019).

2.4 Programming Robot Behaviour

One extension of this thesis may aim to supplement a robot's behaviour, perhaps initially through a form of teleoperation (see Section 2.4.5); the consequents of satisfied rules in the knowledge-base system may contain *actions* (see Section 3.3.4.3), which can be parsed for reference to an intermediate form of a control language, such as the interface provided by the Robot Operating System (ROS) framework (Robotis, 2018).

> "*ROS is a flexible framework for writing robot software. It is a collection of tools, libraries, and conventions that aim to simplify the task of creating complex and robust robot behaviour across a wide variety of robotic platforms.*" (Robotis, 2018)

Although the subject of possible future work, it is of interest to briefly determine the current state of how robot intelligence is defined.

2.4.1 Robot Intelligence

Brooks (1990) refers to and defines two hypotheses of intelligence used as basis for methodologies in building intelligent robots – the symbol system hypothesis and the physical grounding hypothesis.

In other words, according to Brooks, when representing knowledge, two approaches can be taken – the *classical*, purely symbolic approach, or the *reactionary*, physical grounding approach.

2.4.1.1 Symbol System Hypothesis

Newell and Simon (1976) defined the notion of a *physical symbol system* in their Turing Award paper as:

"*…a machine that produces through time an evolving collection of symbol structures*".

They then introduce their hypothesis that a physical symbol system possesses the means for general intelligent action. In other words, intelligent actions rely on the ability to store and manipulate symbols. Brooks (1990) interprets the hypothesis as:

"*The symbol system hypothesis…states that intelligence operates on a system of symbols. The implicit idea is that perception and motor interfaces are sets of symbols on which the central intelligence system operates. Thus, the central system, or reasoning engine, operates in a domain independent way on the symbols. Their meanings are unimportant to the reasoner, but the coherence of the complete process emerges when an observer of the system knows the groundings of the symbols within his or her own experience.*"

Brooks' (1990) criticism of this approach is that although it has had great success in searching, reasoning and playing chess and logic theorem proving, symbolic systems suffer from the frame problem (Hayes, 1981) i.e. the system cannot assume anything that is not explicitly stated, the symbolic descriptions of the world are domain-specific and symbolic systems become overly complex as they incorporate more and more of the chaotic nature of the real world.

2.4.1.2 Physical Grounding Hypothesis

The physical grounding hypothesis was put forward as both a criticism of the symbolic system hypotheses and as an alternative proposal (Brooks, 1990). It was called *nouvelle AI* and it stated that to build intelligent systems its knowledge needed to be associated with the real world:

"*…This hypothesis states that to build a system that is intelligent it is necessary to have its representations grounded in the physical world. Our experience with this approach is that once this commitment is made, the need for traditional symbolic representations soon fades entirely. The key observation is that the world is its own best model. It is always exactly up to date. It always contains every detail there is to be known. The trick is to sense it appropriately and often enough.*" (Brooks, 1990)

Brooks (1990) and others e.g. Matuszek (2015), Richards et al. (2019), non-surprisingly champions physical grounding. Their basis for doing so includes the criticisms of symbolic systems already mentioned, as well as the facts that practical, non-laboratory systems are embedded in

the real world and a symbolic centralised control system requires symbols. Sensor inputs must be converted to symbolic form, itself a difficult task.

Criticism of the physical grounding hypothesis is that physically grounded knowledge does not necessitate intelligence as the systems derived using this hypothesis (see the subsumption model in Figure 2.8 in Section 2.4.2), do not understand what they are doing, they are just reacting to the world around them – they have been coined "mindless machines" (Pollack, 2006). Pollack (2006) refutes this criticism and embraces the moniker, pointing out symbolic systems do not actually *understand* the symbols they are manipulating. He points to emergent, intelligent behaviour inherent to natural systems that lack symbolic reasoning (or even brains) and goes on to say:

> "*Most of what our brains are doing involves mindless chemical activity…*"

The definitions of symbolic and physical grounded systems inform as to how a robot is built and it behaves. Hybrid approaches are possible where a robot can take a reactive subsumption at lower layers, and a symbolic approach for the higher planning layers (Brooks, 1991). In either case, methods must exist on how behavioural knowledge can be imparted to these systems, either by the actual engineered sensor-logic-actuator design of a low level individual layer, or the programming language logic of higher layers. In both cases, it is an extremely difficult task, especially for non-programmers or non-engineers. This is exacerbated by a lack of standardisation or conformity in programming languages, for example, Bonner and Shin (1982) identified the ad hoc manner in which robots were programmed – anecdotally a language was developed for each different platform.

Contemporary mobile robots are either self-contained, executing heuristic algorithms or aspects of formal symbolic logic directly in their hardware, or their cognitive abilities are externalised via networking to more powerful processors executing the heuristics and sending control commands to the hardware and receiving sensor data in return (Ardiny et al., 2015). Teleoperation by a human using remote control can also be the latter, with a human in effect using heuristics, interpreting the autonomous system's environment and issuing commands for the system to action. A combination of autonomy and initial teleoperation (by natural language control) through to behavioural procurement is an aspirational scenario considered for the extension to this thesis.

2.4.2 Hierarchical Layered Behaviour

Analogous with standard software engineering and networking protocol stack approaches, control of a robot can be of the form of a layer-based paradigm (Brooks, 1986). Brooks adopted the philosophy that robots can be controlled by decomposing the control problem into distinct cooperating, hierarchical behaviours of increasing complexity as opposed to separate functional modules as an example of the physical grounding hypothesis. Additional, more complex behaviours can be added incrementally, with higher layers overriding the outputs for actuators of lower layers.

The functional decomposition of the control architecture (perception, modelling, planning, task execution, motor control) is instead decomposed into a task-achieving behaviour layered model he terms a *subsumption model* (Figure 2.8).

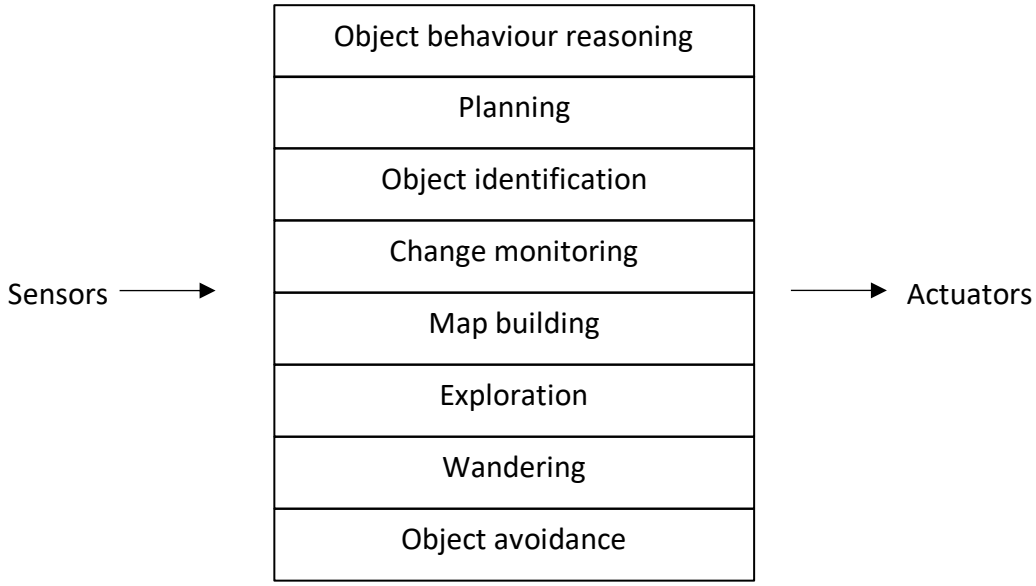


Figure 2.8: Behavioural layers

> *"There is no need for a central control module of a mobile robot. The control system can be viewed as a system of agents each busy with their own solipsist world."* (Brooks, 1986)

Each layer in the model is implemented on separate, physical processors acting as a finite state machines (FSM) communicating asynchronously with each other. The subsumption model can be used as a design philosophy for determining how to impart a robot's behaviour.

Considering then how to impart behaviours to a robot, two common approaches include *Learning from Demonstration* (Chernova and Thomaz, 2014; Fischer et al., 2016) and *Reinforcement Learning* (Kober et al., 2013; Johannink et al., 2019), and deep-learning approaches (not detailed here) include *Recurrent Neural Networks*. All approaches share a common ground, namely to define a hierarchical representation of knowledge that can then allow new skills to be derived from existing skills.

2.4.3 Learning From Demonstration (LFD)

Learning from demonstration (also known by a range of terminologies, such as Learning by Demonstration, LbD, Programming by Demonstration, PbD, Assembly Plan from Observation, behavioural cloning, imitation, and so on), is a type of supervised learning – a machine learning technique that learns mappings between the world state and an associated robot action by the presentation of labelled training data (Argall et al., 2009; Chernova and Thomaz, 2014; Fischer et al., 2016). In contrast to reinforcement learning, LfD learns from repeated examples provided by a demonstrator which are recorded sequences of state-action pairs that are then used by the particular LfD algorithm to determine a reproduction of the demonstrated behaviour. In effect the LfD algorithm is given a labelled set of training data, and it attempts to learn an approximation of the function that created the data.

The traditional approaches to LfD model the dynamics of the domain and produce mathematically-based policies, for example the modelling of infant cognition in a Map-based Encoding algorithm (Pierris and Dahl, 2017). These models, though concise and mathematically sound, are very difficult to derive as they are modelling continuous and complex motor skills and their dynamics (trajectories), may suffer accuracy issues due to approximations, for example, linearisation (Argall et al., 2009), which degrade their performance in the real world domain.

Another problem associated with LfD methods is the *correspondence problem* (Dautenhahn and Nehaniv, 2002). Correspondence basically refers to the differences between the demonstrator and the learner (the imitator). An observer must identify a mapping between the demonstrator and imitator – if they are similar (for example, similar physiology) then the correspondence is obvious (for example, left arm maps to left arm, right foot maps to right foot etcetera), and associated actions also correspond (for example, "*raise left leg*"). But if the embodiment is dissimilar, then the mapping becomes more problematic. For example, how does a robot arm with ten degrees of freedom map to a human arm with seven degrees of freedom? Dautenhahn and Nehaniv (2002) define the correspondence problem as:

> "*Given an observed behaviour of the model, which from a given starting state leads the model through a sequence (or hierarchy) of subgoals – in states, action, and/or effects, while possibly responding to sensory stimuli and external events, find and execute a sequence of actions using one's own (possibly dissimilar) embodiment, which from a corresponding starting state, lead through corresponding subgoals – in corresponding states, actions, and/or effects, while possibly responding to corresponding events.*"

Correspondence mapping is problematic when dealing with real robots (as opposed to simulations) as robots exist in the real word, execute real actions and are subject to physics; the demonstrator thus must demonstrate actions in this environment, and capturing the data associated with these actions requires careful consideration – for example, are sensors placed on the demonstrator, the robot, or an external camera, or is the robot controlled remotely via teleoperation? This is categorised and summarised in Table 2.9 (Argall et al., 2009). The LfD domain is divided by whether the teacher's examples are using the robot directly, or whether they are observed.

Table 2.9: Summary of correspondence mappings

		Embodiment mapping	
		None	Mapping
Record mapping (from Teacher)	None	*1. Teleoperation*	*2. Sensors on Teacher*
	Mapping	*3. Shadowing*	*4. External Observation*
		Demonstration	**Imitation**

2.4.3.1 Teleoperation

Teleoperation is where a robot is remotely controlled by a teacher, so the teacher's states and actions (synonymous with the robot) are directly recorded and the robot can use the recorded data with no further mapping.

An interesting use of teleoperation in the LfD approach is that of a mixed human/robot soccer game (Browning et al., 2004; Argall et al., 2006). Here they combine teams consisting of humans and robots using the same drive platform, a Segway (Segway, 2018). The robot's Segway version is identical to the human version, which addresses the correspondence issue for the locomotive component of the system when the robots are remotely controlled for recording training data. Low-level skill control policies are learned by example trajectories provided by a human operator.

These learned trajectories are then combined with on-board perception abilities in the form of vision (colour object recognition) and tracking.

Esmaili et al. (1995) describe mimicking or "cloning the behaviour" of the skills of a human operator by learning through interacting. Robot navigation can be considered the combination of related sub-tasks, which the authors refer to as behaviours. Their research focuses on a behaviour induced from machine learning techniques to allow a robot to autonomously avoid obstacles – a mobile (wheeled) robot is guided by human operators through simple obstacle courses. Sensor data (infra-red sensors for obstacle detection and motors speed and direction) is collected from at least 300 sets of training sessions with different operators and obstacle scenarios. The data is processed by two induction algorithms (*Induct*, and *C4.5* with extensions to produce *if-then* statements) for comparison purposes to automatically create rules for RDR rule bases. The intent is to use the resulting decision trees when the robot is autonomous and evaluate its skill in avoiding obstacles without human intervention.

2.4.3.2 Shadowing

The shadowing technique for LfD involves robots shadowing or mimicking a teacher's demonstration, but it records the associated sensor data itself (i.e. directly on the robot). This requires a correspondence mapping as the robot is indirectly recording the teacher's actions (its mimicry approximates the teacher's actions). There is extra effort involved as the robot must track the teacher. Nehmzow et al. (2007) use shadow demonstration for a mobile robot to follow walls – the robot observes a human's trajectory, which it then follows, logging its own perception along the path – this is then used to derive a navigation model.

2.4.3.3 Sensors on teacher

This approach is where a teacher is recording motions directly (based on sensors directly on the teacher, such as full sensor-suits). The teacher's demonstration (perhaps as a one-to-one correspondence of joint angles) is observed by a robot (for example via camera images), and so the robot must transform the recorded data into a form so it can understand the exact actions (thus requiring a correspondence mapping of the recorded data). An example is the HOAP-3 robot being taught referee head and arm gestures for basketball (Calinon and Billard, 2007; Calinon et al., 2007).

2.4.3.4 External Observation

The teacher's demonstration is externally observed, but the embodiment of the teacher differs to the robot, so it must transform the recorded images to actions it can understand, and then map the demonstration to its frame of reference (its embodiment). This is the most general approach but it is less precise and reliable than the other methods. Atkeson and Schaal use external observation to balancing a pole upwards (an inverted pendulum). Human demonstration is captured via stereo video, and the robot performs the same task by observing itself using the same camera that observed the human demonstrator (Atkeson and Schaal, 1997*a*,*b*).

2.4.3.5 Learning From Demonstration Summary

Argall et al. (2009) give an interesting summary for the majority of LfD approaches (in this context, a *policy* is the mapping between world states and robot actions):

> "*Non-robotics-experts may be increasingly presented with opportunities to interact with robots, and it is reasonable to expect that they have ideas about what a robot should do, and therefore what sort of behaviours these control algorithms should produce. A natural, and practical, extension of having this knowledge is to develop the desired control algorithm. Currently, however, policy development is a complex process restricted to experts within the field.*"

This opportunity in other words, is that these methods are often too hard for non-experts, requiring complex models and sophisticated and highly technical setups for demonstration data capture, often requiring considerable repetition and a large corpus of capture data. Some of the LfD approaches attempt to simplify models and they do not require expert knowledge of domain dynamics, but it is not always achieved. This suggests another opportunity for non-experts to derive these behavioural actions based on world state by other means (for example, via a fast, simple knowledge acquisition technique such as C-MCRDR).

2.4.4 Reinforcement Learning (RL)

The reinforcement learning approach derive world state mapping to actions by defining some feedback function that gives a reward based on the *desirability* of visiting particular states (Kober and Peters, 2009; Kober et al., 2013). Unlike LfD, RL learns policy through experience, where each state must potentially be visited through trial and error. The reward function must be carefully defined, itself a difficult task, and an actual robot must visit states to receive the

reward (and thus the reinforcement), which may be problematic in the real word (Argall et al., 2009). An alternative to an explicit reward function is to use humans to provide discrete feedback (Loftin et al., 2014; Saunders et al., 2018; Torrey and Taylor, 2013) – the absence of feedback can be implicitly defined as negative feedback, or inversely, if only negative feedback is given, then its absence is positive. Saunders et al. (2018) argue human intervention feedback is crucial, especially when there is potential for serious harm, such as self-driving cars performing badly, or other forms of harm that may result from offensive behaviour like racist and sexist comments from a conversational agent, however, defining enough feedback information, especially in a timely manner, is also difficult.

2.4.5 Robots and NL

If the thesis research is extended by using NL to supplement a robot's behaviour through a rule-based approach (with the developed C-MCRDR CA authoring system itself extended to provide a platform that can accommodate multiple end-point services through conversation), the "granularity" of this rule-based behavioural supplementation might range from:

1. individual, atomic control instructions to activate components, for example, servo motors and sensors in real time;
2. higher-level block behaviour linking (for example, "move" and "turn"), in which the system has semi-autonomy in executing these tasks accordingly without need of intervention (Suddrey et al., 2016) – but external heuristics are needed to determine task sequencing, for example, the coordination of unmanned aerial systems (UAS or drones) in construction (Munoz-Morera et al., 2015);
3. goal-based control ("take out the trash"). This is the purely autonomous state and is the most difficult to achieve. The system must be grounded or aware of its environment and react to environmental changes to achieve the directed goal without continual direct human intervention (Matuszek, 2015).

An approach for parsing NL phrases for autonomous robot control is to map NL to formal, domain-specific languages that are directly parsed by a robot (Kate et al., 2005; Kuhlmann et al., 2004). In this example the researchers transform NL phrases using deterministic grammar rules to the formal robot soccer coaching language, CLANG (Chen et al., 2018), as well as a database query language GeoQuery (Zelle and Mooney, 1996) using transformation rules. Their Semantic Interpretation by Learning Transformations (SILT) system's parsing is achieved using a pattern-matching string approach against rule templates, or a tree-based approach using

words and syntactic (Part of Speech) markers (which encode grammatical classes and morphological information) as the patterns to match against syntactic parse-trees of the original NL sentences. Training is achieved by a set of NL sentences paired with their formal grammar representations, the result being the induced transformation rules. Competing generalised candidate rules generated from the NL parse are ranked using a goodness measure to determine the final candidate rule. A similar idea is presented by Matuszek et al. (2013). Here they define a high-level LISP-like language, Robot Control Language (RCL) that is produced from a probabilistic parse of NL expressions. RCL instructions are then sent to an execution system that targets the specific hardware. RCL instructions are associated with route traversal. Their goal was "*... to evaluate the ability of the system to generate RCL command sequences that allow a robot to navigate through a labeled map according to NL instructions.*" (Matuszek et al., 2013; Matuszek, 2015).

This robot-specific control language concept could be adopted by the C-MCRDR CA system, however, instead of targeting proprietary, closed robotic platforms, or even designing a specific physical robot architecture, an *off the shelf* platform (such as Robotis' TurtleBot3 (*Turtlebot3*, 2020)) that leverages a huge community-based effort for development of standards of drivers and programming languages could be used.

There is however a trade-off between providing explicit control, giving a robot precise step-by-step instructions via NL, but ensuring the system is grounded (via human perception), or allowing the robot to have some autonomy, at the expense of behavioural brittleness when the domain changes (Ardiny et al., 2015). Ardiny et al. (2015) stated:

> "*In order for robots to be eventually used in fully automated construction sites, there is a need to adopt more sophisticated decision-making techniques that treat autonomous construction with the richness that it deserves.*"

Suddrey et al. (2016; 2017) present an approach to learning complex hierarchical tasks through natural language instruction. They attempt to overcome the challenge of having to retrain the robot in novel situations by generalising learnt tasks. By this they mean tasks can be applied to different objects and scenarios to those used during the learning stage.

In the medical domain, robots have been used as sensor-enabled interactive interfaces (D'Este et al., 2008). This introduces a robot to be used for Medication Review (MR). The robot contains a variety of sensors (touch screen, touch sensors, camera and a microphone) and can be extended with external USB sensors for data related to blood pressure, temperature, blood sugar, and pulse. The system includes a conversational agent that relies on an MCRDR knowledge-base

to ask a patient questions about their medication via text-to-speech. The patient can reply via the touchscreen, touch sensors and speech. The resulting raw sentences are sent to the conversational agent for parsing, resulting in cases sent to the MCRDR knowledge-base for classification. Other sensor readings can also be sent to the knowledge-base for inference. In either case, conclusions are referred to the conversational agent to parse into more natural language style sentences which are then sent to the robot to speak to the patient.

Hybrid systems can allow linguistic control, but with feedback from sensor data to allow adaptation to the environment (Zemalache and Maaref, 2009). Their drone navigation control system uses a self-tuneable fuzzy inference system (STFIS), which is a collection of multilayered neural networks, each designed to control aspects (e.g. motors) of a flyer engine called XSF.

Anticipating the natural language input in a probabilistic manner is another approach (Tellex et al., 2011; Kollar et al., 2010). This research uses the concepts of a Generalized Grounding Graphs framework. Here, a natural language command's context in terms of its hierarchical and structural composition are considered, with the corpus of commands used for training being obtained from crowdsourcing. The resulting model from an NL phrase is a mapping (or grounding) from the command to some aspect of the world. Aspects include objects, places, paths or events. For example, navigational landmarks like "*the desk*" or "*the computers*" are grounded into the robot's model by using a database of tagged images. NL phrases that are spatial in nature are then associated with a robot action in the form of a navigation or directional understanding. The NL input is parsed to a sequence of spatial description clauses (SDCs), which are then combined with a semantic map of the robot's environment to plan a navigational path to an endpoint.

As previously discussed, the task of designing an architecture for robot control and interaction is complex. The ideal goal would be for the robot to be fully autonomous and self-contained, but allowing for external human interaction and/or advice. The architecture can be designed to allow for domain independence, which maximises flexibility at the expense of fine-level domain specificity, or it can be domain-dependent, maximising its applicability (Arkin, 1995; Brooks, 1990). Obviously, the robotic hardware itself also has implications for the domain in which it is used. For example, a robot following advice from a neurosurgeon has a control system designed to be very domain dependent, but the robot cannot sweep the floor. Equally well, you would not want nor expect a room-exploring robot sweeper to operate on your brain.

2.4.6 RDR and Robot Interaction

The possibility of RDR (or its derivations) as an underlying rule-based technique to interact with end-point systems such as robots is an exciting avenue for future research as there are few examples in the literature of work in this area. Although this is not directly related to the research questions being addressed, the overarching principle concept of conversational-based platform that can accommodate disparate services is still relevant. A short review to determine RDR's potential in this area follows..

One approach that uses an RDR variant is that by Shirazi and Sammut (2008). They take the learning by example approach (also known as behavioural cloning – please see Learning From Demonstration (LFD) in Section 2.4.3) to use a variant of RDR, Dynamic RDR (DRDR), to learn to fly a Cessna plane in a flight simulation program. They perceived the issue in a rapidly-changing complex dynamic system is that an expert is making sub-cognitive decisions so quickly that they cannot articulate or justify their decision process during or even after the events where they occurred. In this case, an LfD approach can be used to record a pilot's actions. These actions can then be used as inputs to an induction algorithm to produce ripple down rules automatically.

In prior but related work (Sammut et al., 1992), flying a plane in a simulator relied on the induction of rules via Quinlan's C4.5 induction algorithm (Quinlan, 1993). C4.5 was later replaced with their own induction algorithm, DRDR (Shirazi and Sammut, 2008). This was because the C4.5-derived decision trees were large and unreadable. The authors also indicated the induction stage required many labelled cases to train, and the derived decision tree resulted in brittle behaviour (the system could not cope with moderate environment change). The DRDR system described (Shirazi and Sammut, 2008), needed to produce four conclusions for a given simulator state (elevators, flaps, ailerons, throttle) but instead of using MCRDR, they chose to combine four separate knowledge-bases and coordinate the output by their modified DRDR algorithm.

To reconstruct the flying skill, they record the pilot's performance and analyse the data. The data associated with many trials of different pilot's actions in seven stages of flight (taking off, level flight, turn right, turn left, line up with runway, descend, land) is logged, but in this study, the pilot can maintain the DRDR knowledge-base through a source-code change of the flight simulator. This maintenance requires the simulation to be stopped.

When the pilot cannot articulate how they are arriving at a decision (for example, why they are changing an aileron's pitch), they switch back to the simulation and demonstrate what they

would do – the flight data is logged and DRDR rules are then constructed (via induction) using Learning DRDR (LDRDR). LDRDR is introduced to deal with sequential data found in the log files (in other words, to deal with contextual data, such as the aileron's slight change in pitch compared to its previous value) instead of other machine learning techniques for constructing RDR rules, such as Induct/RDR (Gaines and Compton, 1995) (which is batch-based). Rules are then constructed by the attributes that are found in the log file that caused an action.

The main intent of the LDRDR system is to show the feasibility of capturing or cloning sub-cognitive behaviour using a combination machine learning and knowledge acquisition (Shirazi and Sammut, 2008). Although the system demonstrates RDR and their variant of Induct RDR (LDRDR) as a method of providing this behaviour, it is applied to highly dynamic systems and is not, as such, a method for a non-programmer to impart arbitrary behaviours to an autonomous system. Their system shows how machine learning and knowledge acquisition can be used to learn to control a dynamic system, which requires considerable time for repeated examples (demonstrations) and consequently, the equivalent of labelled data used for induction. What is interesting is how RDR-imparted knowledge can be applied to simpler systems in a direct manner, without prior learning, and how this knowledge can be procured from other existing system knowledge or from the domain user.

As mentioned earlier in Section 2.4.3.1, Esmaili et al. (1995) used RDR as an acquisition technique, with the rule-base inductively populated by Quinlan's C4.5 induction algorithm (Quinlan, 1993) and Induct/RDR (Gaines and Compton, 1995) for training object avoidance behaviour using teacher demonstration (LfD). This approach appears to be refined by the research associated with the flight simulator mentioned above where C4.5 is replaced by their own induction algorithm, LDRDR.

In their research on extracting features from range images, Sheh et al. (2007) take a learning by example approach to generate a terrain model from repeated image data of a "stepfield" terrain (a block-based terrain of various heights) that is processed using computer vision and digital image processing techniques. They use edge (and height) detection to determine the terrain a robot is encountering. These models are produced after repeated movement of the robot in its environment. In deployment, RDR is used to train the robot by determining positioning based on terrain feature detection inputs that are comprised of a range image and heading-attitude sensor measurements. The operator can override the trained responses by creating a new rule in the RDR knowledge-base. To summarise, the work uses RDR to learn human expertise to recognise terrain features, an LfD approach.

The idea of interpreting RDR classifications as actions was implemented by Burman et al. (2006) in their game character AI behaviour maintenance approach. The researchers attempted to provide an environment for game players to personalise or modify the in-game non-player-character's (NPC) behaviours (for example, team-based games where the player controls groups of characters trying to achieve a 'capture the flag' goal). They modified MCRDR to only give one classification per inference cycle (which appears to be due to the codebase they used for development, otherwise RDR could have been used) as the AI characters could only decide on one action (out of 8) at a time. Conversely, each conclusion contained a set of actions, and one action was chosen randomly to give a degree of unpredictability of the behaviour. Each in-game event causes the NPC to seek an appropriate action from their MCRDR knowledge-base. The system was evaluated with 60 participants creating (editing) RDR rule-bases, and on average, each participant created 15 rules, but the rules tended to be limited to simple behaviours. The authors determined the issues were due to the apparent difficulty participants had in determining rule conditions (features) and the constraints caused by confusion with the user interface.

The work by Burman et al. (2006) is very interesting as it provided a means of exposing the description of behaviour for the NPC characters to real players, using (in effect), RDR. The behaviours are fixed by a small group of eight axiomatic primitive behaviours, but as the game is a simulation, the world events and actual player actions are extremely contrived and not grounded in any real domain. Also of interest here is there are no correspondence issues or concerns that are addressed as they simply don't exist – this is not the case in any real-world domain. The system is also deficit in having to concern itself with external sensing.

With this in mind, RDR has also been used to classify a robot's sensor readings, for example, vision, to assist with object recognition (Pham and Sammut, 2005; D'Este and Sammut, 2008). These systems use ranged numerical data from camera sensor readings for vision recognition. The system used by Pham and Sammut for a soccer-playing robot uses an RDR variant, *Cut95* (Scheffer, 1996) to allow for interval-bounded data attributes. The robot's camera provides brightness and chrominance information (YUV) for each pixel, with each channel ranging from 0 to 255. A tuple containing pixel attributes $< x_1, x_2, \ldots, x_n >$ satisfies an RDR rule's antecedent if each value falls within the antecedent's set of intervals, $[m_1, M_1], [m_2, M_2], \ldots [m_n, M_n]$ where each m_i is the lower bound and M_i is the upper bound of an attribute x_i. The approach adopted by D'Este and Sammut (2008) uses the *Clarify* system which extracts pixel YUV attributes from the robot's camera via a built-in motion segmentation algorithm. Each feature (colour, shape, size, weight and position) has its own RDR

knowledge-base for classification, and during training the feature classification name is given by the domain expert via speech.

RDR has been used for robot action planning (Kwok, 2002), where the authors divide up the state space into regions, each region has an associated RDR rule and conclusion. As a robot enters a region, the RDR rule associated with that region fires, with the system repeating inference and splitting regions to allow new actions to be described by the expert.

Defining or controlling a robot's behaviour is a difficult task, especially for non-experts. For environments where there are no highly-dynamic physical activities (for example, throwing and catching a ball where anticipatory actions require a high degree of sensing and modelling the physics involved, such as ball's ballistic trajectory), a possibility then exists that higher-level behaviours (consisting of the linkage of well-established fundamental behaviours) could be defined and provided by a knowledge-base system – which needs to overcome the knowledge acquisition and maintenance bottlenecks, which is where C-MCRDR fulfils this requirement. The last factor to consider is how is this knowledge acquired by the KBS from non-experts, the conversational authors. A natural language interface, in terms of providing an avenue for knowledge acquisition in a natural manner, and a means of 'conversing' with a robot, is a logical progression. Humans also converse with topical context, which is assumed when communicating with others – so the design of a conversational agent system, which retains conversational context would be advantageous as a natural way to instruct a robot.

Chapter 3

Methodology

"*In ancient times cats were worshiped as gods; they have not forgotten this.*" – Terry Pratchett (Rainbolt, 2007)

This chapter contains the following sections:

3.1 Introduction

This section contains the following subsections:

The overall methodological approach to the research is detailed in Section 3.2. To provide a context for the methods used, the research questions that are being answered by the thesis are repeated here (from Section 1.2.1), and the data-needs summary for each sub-research question is detailed in this section.

3.1.1 Research Questions

Research Question (RQ): *How can a human-authored rule-based knowledge-base system be defined and applied to create a conversational agent system that can achieve high levels of system performance without relying on complex scripting and programming skills or knowledge of formal grammatical syntax to specify the conversational knowledge?*

Sub-Research Questions (SRQ):

SRQ1 - *How can conversational context be maintained in a human-authored rule-based conversational agent system?*

SRQ2 - *What is a pattern-matching approach that does not require complex scripting and programming skills or knowledge of formal grammatical syntax that can provide high levels of system performance?*

SRQ3 - *What augmentations to the MCRDR rule-based KBS methodology will produce conversational agent systems that can achieve high levels of system performance?*

SRQ4 - a) *What is the best current market-leading Intelligent Personal Assistant (IPA) to leverage as a speech component of a conversational agent system in terms of ASR performance that allows the default vendor-defined conversational agent to be supplanted?*; and

b) *How can the IPA's associated ASR errors when coupled to a human-authored rule-based KBS be corrected?*

3.1.2 Summary of Data Needs, Methodologies and Evaluation Plan

The associated data requirements for each research question are detailed in Table 3.1.

Table 3.1: Methodology data requirements

SRQ	Data Needs	Method	Evaluation
SRQ1	MCRDR Source code; User interaction data via system log files from C-MCRDR CA system evaluation	Propose modification to the original MCRDR inference algorithm resulting in a new methodology, called Contextual MCRDR (C-MCRDR) to start inference with the assumption that the rules previously satisfied in the last inference request are also satisfied for the current inference request - refer to Sections *3.3.1 Inference Modification – Topical Conversational Context* and *3.3.11 C-MCRDR CA System Implementation*. Design an evaluation trial, including participant recruitment, Ethics approval - refer to Section *3.3.13 C-MCRDR CA System Evaluation Methodology*	Implement a C-MCRDR conversational agent system for a pedagogical domain and evaluate the system's performance in terms of appropriate responses and user acceptance via an integrated feedback system. The user's perception of the validity of the system's responses to their conversational requests (questions) especially when related to conversational depth (which is a measure of the maintenance of topical conversational context) can be ascertained. Refer to Section *4.2 C-MCRDR CA System Evaluation*.
SRQ2	C-MCRDR Source code; User interaction data via system log files from C-MCRDR CA system evaluation (as per SRQ1)	Modify the C-MCRDR methodology to use a constrained pattern-matching, lexical paraphrasing approach in order to simplify conversational knowledge acquisition and maintain a low cogitative overhead for the non-expert conversation author. Refer to Sections *3.3.2 Dictionary/Lexical Paraphrase Approach*, *3.3.11 C-MCRDR CA System Implementation* and *3.3.13 C-MCRDR CA System Evaluation Methodology*.	As per SRQ1, implement and analyse system performance and participant evaluation. The user's perception of the validity of the system's responses to their questions especially when associated with a simple lexical paraphrasing interface can be ascertained. Refer to Sections *4.2 C-MCRDR CA System Evaluation* and *Section 4.2.10 User Acceptance (Feedback)*.

Continued on next page

Table 3.1 – *Continued from previous page*

SRQ	Data Needs	Method	Evaluation
SRQ3	C-MCRDR Source code; User interaction data via system log files from C-MCRDR CA system evaluation (as per SRQ1)	Modify C-MCRDR algorithm to capture and retain pre-established context via a suitable interface for inter- and intra-topical context - refer to Section *3.3.4 Context Variables*. Investigate a post-inference method to facilitate **Natural Language Interfaces to Databases** (NLIDB) in order to reduce rule-bloat and subsequently decrease rule maintenance (conversational outputs) overhead. Refer to Sections *3.3.6 Post-Inference* and *3.3.6.1 Database Querying*. Implement a method for brittleness mitigation - refer to Section *3.3.8 Brittleness Mitigation*. If coupled to speech input and output, determine methods to pre-process speech input (correcting for pre-assumed textual forms, for example, dates, numerical ranges etc) and pronunciation problems when speech synthesis is performed. Refer to Sections *3.3.9 Speech Considerations*, *3.3.11 C-MCRDR CA System Implementation* and *3.3.13 C-MCRDR CA System Evaluation Methodology*.	As per SRQ1, implement and analyse system performance and participant evaluation. The effectiveness of an NLIDB interface, coupled with the associated rule count reduction and subsequent rule maintenance overhead minimisation will be determined. Refer to Section *4.2 C-MCRDR CA System Evaluation*. Conversational knowledge acquisition by *non-experts* - inclusion in two external research projects where the conversational authors do not possess knowledge of scripting or knowledge of formal grammatical syntax - clinical training via a 3D dialog simulation (Yang et al., 2019), and a task-oriented language assistant for improving English as a second language (Gu et al., 2019).

Continued on next page

Table 3.1 – *Continued from previous page*

SRQ	Data Needs	Method	Evaluation
SRQ4	C-MCRDR Source code Evaluation word lists and associated spoken frequency (via British National Corpus dataset); Evaluation sentence lists (via software-generated text sourced from a copyright-free novel); IPA device ASR results from C-MCRDR conversation system log files	Measure the IPA device's response to pre-determined speech input, determining attributes of the input that influences the success (or otherwise) of the ASR output. Refer to Sections *3.4 Intelligent Personal Agent Methodology* and *3.4.2 Recognition Performance Improvement*. Correct the results of ASR errors via two proposed methods – Global ASR correction and Regional ASR correction. Refer to Sections *3.4.2.1 Global ASR Correction* and *3.4.2.2 Regional ASR Correction*.	Two widely available IPA devices, Google Home and Amazon Echo are evaluated by coupling them to a sparsely populated C-MCRDR knowledge-base. ASR performance is measured and the Global ASR correction method performance is evaluated in terms of the number of evaluation rounds needed. Refer to Section *4.3 Intelligent Personal Assistant Evaluation*.

3.1.3 Knowledge Acquisition

As detailed by the literature review in Chapter 2, Multiple Classification Ripple Down Rules (MCRDR) is chosen as the initial knowledge-base system in this research. MCRDR is a knowledge representation, inference and acquisition technique (Kang, 1995) to allow a knowledge-base system (KBS) to be easily maintained without requiring a Knowledge Engineer. This means a domain expert can rapidly classify attribute-based input cases without needing to have in-depth technical skill, and they can concentrate on what they know best – their domain expertise. MCRDR by its very nature provides rapid, automatic data validation, an important aspect that maintains the integrity of the system without high cognitive and computational overhead.

3.2 Overall Methodology

This research was conducted in two logical phases, each of which directly addresses the sub-research question aspects of the research problem – the first and main phase alters MCRDR and evaluates the derivation, C-MCRDR to support conversational context with constrained natural language input case data, and the second phase determines the best, current market-leading interface (an Intelligent Personal Assistant (IPA)) to couple with the developed system, as well as providing and evaluating a method to correct ASR errors exhibited by the chosen IPA. The two phases are:

- *C-MCRDR CA* – this phase addresses research questions SRQ1, SRQ2 and SRQ3 (see Section 1.2.1), and it included the design and implementation of a natural language conversational question and answering system using a C-MCRDR knowledge-base system as the underlying technology. The developed C-MCRDR CA system was then evaluated to validate the suitability of the approach (see Section 3.3.13 for the evaluation methodology and Section 4.2 for the evaluation). The specific methodology for this phase is detailed in Section *3.3 C-MCRDR CA System Methodology.*
- Intelligent Personal Agent – this phase evaluates contemporary, commodity speech-to-text systems in order to choose the best-performing system as a speech interface for the conversational agent system defined and developed in the previous phase. Research question SRQ4 (Section 1.2.1) is addressed by this phase. The specific methodology for this phase is detailed in Section *3.4 Intelligent Personal Agent Methodology* and the evaluation is in Section 4.3.

3.3 C-MCRDR CA System Methodology

This section contains the following subsections:

The first phase of the methodology incorporated the design, development and evaluation of a C-MCRDR CA system consisting of:

- an end-user chatbot interface with natural language capabilities;
- an external relational database management system;
- a domain expert administrative interface; and
- a modified underlying MCRDR knowledge-base.

The modifications to the standard MCRDR inference mechanism, together with associated support classes and mechanisms detailed later in this section is called *Contextual MCRDR* (C-MCRDR). C-MCRDR can be readily applied to be the basis of conversation systems by

inferring appropriate responses from user utterances. The clear, clean separation of knowledge from the inference engine in a contemporary KBS is analogous to the separation of chat scripts (such as AIML) from a conversational agent program (Compton, 2011; Wallace, 2011). Chat scripts embody conversational responses as a form of knowledge, and it is this knowledge that can be captured and represented in a KBS approach. Inter-domain applicability, commonly criticised for CA and NLIDB systems, can be partially addressed in the C-MCRDR CA system by providing interface support for knowledge-base rule, query, dictionary and ASR correction rule reuse for relevant commonality between domains.

The C-MCRDR approach also takes advantage of the fact that the number of rules and rule redundancy in the knowledge-base can be significantly reduced when rule antecedent and consequent values are able to be generalised – conversational contextual information is then used to bind C-MCRDR inference results to specific classifications. The approach achieves this by the following means:

1. Maintenance of conversational context by adopting a stack-based approach to inference results (see Sections 3.3.1 and 3.3.5);
2. Lexical paraphrasing pattern-matching, the use of dictionaries as a localised ontological approach (see Section 3.3.2);
3. Context-based variable assignment (via regular expression pattern matching) of relevant context that is maintained across inference requests (see Section 3.3.4);
4. Post-inference deferred classification via database query expressions (bound by relevant context variables) (see Section 3.3.6).

3.3.1 Inference Modification – Topical Conversational Context

This methodology is aligned with SRQ1 (see Section 1.2.1): *How can conversational context be maintained in a human-authored rule-based conversational agent system?*

Human chat discourse follows one or more topics during a conversation, the automated detection of which is an ongoing area of research (e.g. TF-IDF approaches, (Adams and Martell, 2008)). Humans naturally maintain an appreciation of what is being discussed without having to continually re-emphasise it. For example, the following shows a stilted, unnatural conversational excerpt:

Speaker1: "The **weather** is going to be bad today. *As for the weather*, the prognosis is for rain."

Speaker2: "What will be the *weather's* temperature?"

Speaker1: "The *weather's* temperature will be 10 degrees Celsius."

A speaker intuitively understands the current topic is *weather* – all subsequent explicit mentions of weather are redundant. To achieve this more natural, contextual response, the C-MCRDR inference algorithm is altered by influencing the starting rules where inference begins in the decision tree. This is achieved this via a stacking of inference results (Glina and Kang, 2010; Mak et al., 2004) – each stack frame contains a set of satisfied rule(s) from the previous inference request. Frames are popped from the stack in LIFO order, and inference simply assumes each rule in the frame's ruleset is satisfied as a starting point to evaluate child rules. For each dialog session, C-MCRDR implicitly assumes each case is part of a temporal sequence (i.e. components of a continuing dialog) and it is not initially independent of other preceding cases. If no child rules are satisfied in the current stack frame, the frame is temporarily discarded and inference is attempted again with the new top of the stack rule set. Eventually, if no further stacked rule sets are available, the default rule is satisfied.

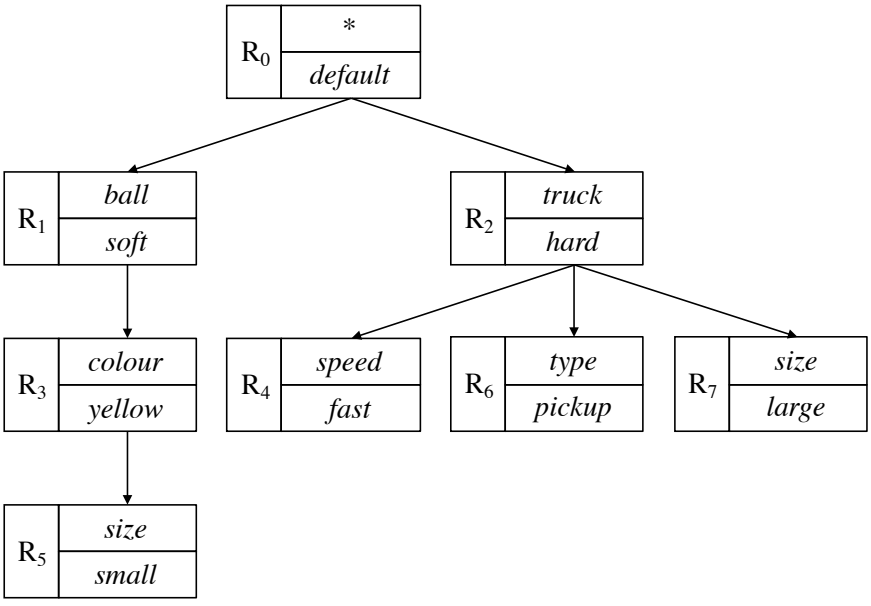


Figure 3.1: Simple knowledge-base

As an example, consider the knowledge-base shown in Figure 3.1. Each node represents a rule in the knowledge-base, with the top section of a node representing the rule's antecedent that simply pattern matches against a case's dialog text attribute by the text `contains` operator.

The bottom section of a node represents the consequent (conclusion or classification) value. Consider the conversation seeking answers using standard MCRDR (rule antecedent matches shown in **bold**) in Table 3.2.

Table 3.2: Standard MCRDR inference

Question	**Satisfied Rules**	**Inference Response**
"How hard is the **ball**?"	R_0,R_1	"soft"
"What **colour** is the **ball**?"	R_0,R_1,R_3	"yellow"
"What **size** is the **colour ball**?"	R_0,R_1,R_3,R_5	"small"

Table 3.3: C-MCRDR inference algorithm

Algorithm

INITIALISE: LET stack $S \leftarrow \varnothing$
FUNCTION C-MCRDR(S,Case c)
    LET stack $S' \leftarrow S$
    LET rule set $R \leftarrow \text{pop}(S')$
    WHILE $R \neq \varnothing$ DO:
        LET $R' \leftarrow \bigcup_{i=1}^{n} I(c, r_i) \forall r_i \epsilon R$
        IF $R' = \varnothing$ THEN
            LET $R \leftarrow$pop(S')
        ELSE
            push(S, R')
            RETURN R'

    LET $R \leftarrow I(c, r_{default})$
    RETURN R

The C-MCRDR inference algorithm as modified by the stacking approach is summarised in Table 3.3, noting the inference function $I(case, rule)$ returns a set of n satisfied rules $\{r_0 \,..\, r_n\}$. It should also be noted frames are never deleted from the inference result stack during a dialog session, and inference result frames only containing R_0 are never pushed to the stack (apart from the first frame). The former constraint in situations where a topic may need to be invalidated is addressed by a do-not-stack metarule, although this was not evaluated in the C-MCRDR CA system (see Chapter 5 *Conclusions and Future Work*) as the evaluated domain did not require topic invalidation. Conversational domains where non-retention of stacked context might be relevant, could, for example, include a domain where a customer is requesting costs for booking multi-city flights where they may want to consider starting the process again, forgetting any previously requested flight information.

Table 3.4: Contextual MCRDR inference

Question	**Response**	**Stack**
*How hard is the **ball**?*	*soft*	$\{R_1\}, \underline{\{R_0\}}$
*What is its **colour**?*	*yellow*	$\{R_3\}, \underline{\{R_1\}}, \{R_0\}$
*What **size** is it?*	*small*	$\{R_5\}, \underline{\{R_3\}}, \{R_1\}, \{R_0\}$
*How hard is the **truck**?*	*hard*	$\{R_2\}, \{R_5\}, \{R_3\}, \{R_1\}, \underline{\{R_0\}}$
*What's **type** and **speed**?*	*pickup,fast*	$\{R_4, R_6\}, \underline{\{R_2\}}, \{R_5\}, \{R_3\}, \{R_1\}, \{R_0\}$
*What **size** is it?*	*large*	$\{R_7\}, \{R_4, R_6\}, \underline{\{R_2\}}, \{R_5\}, \{R_3\}, \{R_1\}, \{R_0\}$

Returning to the conversational example in Figure 3.1 and Table 3.2 using C-MCRDR it now contains a stack of previous inference results. The inference stack rules that were successful in producing a non-default (R_0) response as the starting point for the current inference request is shown underlined in Table 3.4. Note that the stack frame containing R_0 is never re-added to the stack. The conversion can now be more natural, as shown in Table 3.4 as the topic does not need to be repeated.

3.3.2 Dictionary/Lexical Paraphrase Approach

This methodology is aligned with SRQ2 (see Section 1.2.1): *What is a pattern-matching approach that does not require complex scripting and programming skills or knowledge of formal grammatical syntax that can provide high levels of system performance?*

The primary approach in this phase is to pattern-match input dialog patterns to rule-based expressions using a lexical paraphrase database approach, similar to the approach of WordNet (Miller, 1995) and other lexical paraphrase generation systems (Oh et al., 2015; Li et al., 2005; Fader et al., 2013). However, although developmental effort could extend the system to use massive, external lexical databases, a simple, localised (and domain-specific) dictionary is used.

3.3.2.1 Dictionary Text Transformation

Dictionary terms are *greedy* pattern-matched for associated synonyms, in other words, the synonym paraphrase with the greatest number of terms is matched versus phrases with a smaller

number of terms. Synonyms (or phrasal terms) can be multi-word expressions, or general *regular expressions*, which are prefaced by the syntax:

```
/RE:<expression>
```

It should be noted regular expression syntax usually requires a degree of technical skill on behalf of the conversational author (the domain expert), and the developed system's graphical user interface provides some guidance in regular expression construction. Although a rule-based exact string match is being used, approximate string matching methods (Raghavan and Allan, 2005; Ukkonen, 1992; Hakak et al., 2019; Sidorov et al., 2014) such as the string edit distance (Levenshtein, 1966; Hall and Dowling, 1980; French et al., 1997; Zobel and Dart, 1995) could alternatively have been used (particularly when a source utterance is an ASR transcription).

Assume T is the initial text to match (the case text, which is the user utterance) for associated dictionary terms, T' is the text post-match (synonyms replaced by lexical representative terms) and assume D is a set of size n single dictionary terms:

$$D = \{d_i \mid 1 \leq i \leq n\} \tag{3.1}$$

Each d_i has an associated set of synonyms, S_i:

$$S_i = \{s_{ij} \mid 1 \leq j \leq n_i\} \tag{3.2}$$

with the set of all synonyms defined as:

$$S = \bigcup_{i=1}^{n} S_i$$

All synonym phrases must be unique, i.e.

$$\begin{aligned} S_k \cap S_m = \emptyset \forall k, m \mid & 1 \leq k \leq n_k \\ & 1 \leq m \leq n_m \\ & k \neq m \end{aligned}$$

Synonym phrases s_{ij} in T are replaced by their associated dictionary terms, d_i by the dictionary function, D_F:

$$D_F(T) \rightarrow T' \mid \forall s_{ij} \subseteq_s T, s_{ij} \rightarrow d_i \tag{3.3}$$
$$1 \leq j \leq n$$
$$1 \leq i \leq n$$
$$\subseteq_s \textit{ is defined as the substring operator}$$

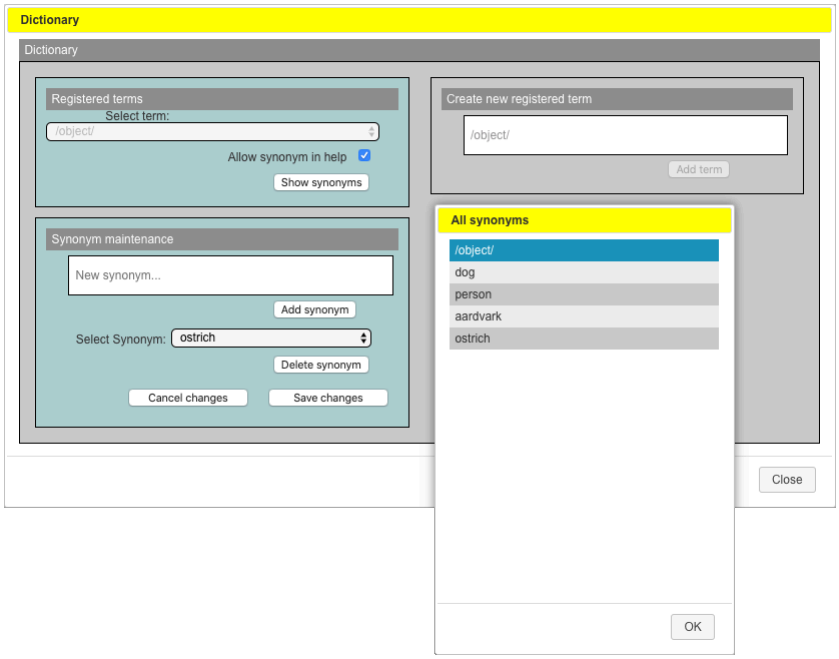


Figure 3.2: Dictionary interface

Table 3.5: Example dictionary terms

Term (d_i)	**Synonyms** (s_{ij})			
	s_{i1}	s_{i2}	s_{i3}	s_{i4}
/object/	dog	person	aardvark	ostrich
/article/	a	an	the	that
/dayofweek/	/RE:[mtwfs]1[ouehra]1[neduit]1[a-z]*day			

The C-MCRDR CA system implementation interface for the dictionary is shown in Figure 3.2. Dictionary representative terms cannot be deleted (due to potential validation and verification issues), but individual synonyms can be edited. Table 3.5 shows some simple dictionary terms aligned with the definitions indicated in Equations 3.1 and 3.2.

3.3.3 Other MCRDR Augmentations

Methodological augmentations that follow are aligned with SRQ3 (see Section 1.2.1): *What augmentations to the MCRDR rule-based KBS methodology will produce conversational agent systems that can achieve high levels of system performance?*

3.3.4 Context Variables

An additional aspect to maintain conversational (or factual) continuity is the retention of context-specific data found in cases as they arise on a temporal basis.

One of the aims of the research is to reduce the number of rules required by the knowledge-base, and this is achieved by allowing partially generalised rule classifications to be generated by the inference engine, which are then finalised post-inference (or bound) by the consequent-parsing component of the implemented system.

As stated in Section 3.3.2, one or more of a rule's antecedent attributes may be satisfied by the presence of lexical (dictionary) terms. Generally, the actual synonyms that matched may need to be retained for future reference and inference. For example, a rule might be satisfied if an utterance contains the term `/animal/`. More specifically, it may be more interesting to determine *which* animal matched, such as `aardvark`.

It is thus a requirement that the specific values of attributes that caused a rule's antecedent to be satisfied must be known. For example, if an utterance mentions *a coloured ball* that satisfies a particular rule, *what* colour is also meaningful as a context for later inference. As a dialog progresses, C-MCRDR can retain this matched data via *context variables*. Context variables (together with system context variables, or *default values*) could be considered as a simplified notion of slots in a frame as first put forward by Minsky (Minksy, 1975), with slots retained (temporally) across all subsequent frames. Variables are defined by the domain expert (guided by a GUI) with simple meta-rules referring to dictionary terms, literal values, or arbitrary regular expressions. The expert can reference variable values in a rule's antecedent or consequent with an XML-style syntax (see Tables A.10 and A.11 in Appendix A), which is also guided by a GUI. Variable assignment can also occur as a side-effect of a consequent (via an *action* – this feature was implemented but not used in the C-MCRDR CA system).

As a simple example of variables, consider the dictionary defined previously in Table 3.5. Any *synonym* in a case's dialog attributes may match an associated dictionary *term*. Rule antecedents are built using a collection of terms (and not specific synonyms) and surrounding string literals. Consider a simple rule antecedent consisting of the following text:

```
/article/ /object/ flies
```

This simple pattern-matching syntax (which is guided by the system's GUI interface) indicates the *article* and *object* terms must be present (using the system's *contains* operator), followed by the literal *flies*. This has been simplified here for brevity as rule antecedents are built via the C-MCRDR CA system's GUI interface, and a more formal representation could be considered:

```
contains(/article/)
literal(' ')
contains(/object/)
literal(' ')
literal('flies')
```

Table 3.6: Example rules – assume R2 and R3 are child rules of R1

	Condition	**Conclusion**
R1	/article/ /object/ flies	<LIT>@object</LIT>s should not fly.
R2	I like pets	You keep <LIT>@object</LIT>s as pets?
R3	/article/ /dayofweek/ flies	<LIT>@object</LIT>s do not fly today

Table 3.7: Example final parsed responses

Case data	**Final parsed consequent**
An umbrella flies	*umbrellas should not fly*
That Thursday flies	*umbrellas do not fly today*
The aardvark flies	*aardvarks should not fly*
I like pets	*You keep aardvarks as pets?*

Table 3.8: Simple context variables

Variable	**Matching Criteria**
@year	/RE:[1-9][0-9][0-9][0-9]
@SYSTEMyear	set(2019)

The case text "*an aardvark flies*" satisfies the rule, but in some circumstances the consequence needs to specifically refer to the actual `/article/` or `/object/` matched synonym(s) in the case text – the final consequence for this example is "*aardvarks should not fly...*", referring to the stated `/object/` from the case dialog text (see Tables 3.6 and 3.7). Note the stacked inference results as mentioned in Section 3.3.1 and Table 3.3 are partially shown in Table 3.7, as the

response to *That Thursday flies* relies on the context of rule R1 having already been satisfied (and the contextual information, *umbrella*, has been captured by a context variable, `@object`).

In order to capture the matching lexical, phrasal paraphrases or literal terms, system-wide meta rules are defined consisting of a variable name and one or more matching criteria (such as dictionary terms, literal values, or arbitrary regular expressions) – see Table 3.8. Any inference request containing the matching criteria binds the associated context variables, and they are retained across subsequent inference requests. Rule consequents are then constructed by the domain expert using a GUI that can include simple XML-style referential syntax for queries and context variable values that are parsed by the system's output parser. The <`LIT`> operator shown in Table 3.6 *Conclusion* is requesting evaluation of the `@object` context variable, which would be `aardvark` in this example (see Table A.11 in Appendix A for the XML syntax).

In the C-MCRDR CA system implementation, context variables can also be used as:

- attributes in rule antecedents i.e. to satisfy a rule, a particular context variable must be defined or have a specific value (this was not evaluated in the C-MCRDR CA evaluation study as the evaluated domain did not require such a feature for demonstrating the rule-based approach adopted. However, more complex, syntactically rich domains are anticipated to need context variables as attributes in antecedents, so this will be a consideration in future work (see Chapter 5 *Conclusions and Future Work*));
- variable assignment as a side-effect of a consequent (via an *action*, which is also evaluated by the output parser)

3.3.4.1 Context Variable Definition

Context variables are defined based on pre-defined criteria, or as a consequence of a rule that is satisfied.

Assume C is a set of user-defined variable IDs:

$$C = \{c_w \mid 1 \leq w \leq c\} \tag{3.4}$$

Individual c_w identifiers may also be defined via *actions* (see Section 3.3.4.3).

In terms of defining a c_w variable *criteria* to match prior to evaluation, the set C' contains each associated variable matching criterion, c'_w, which is the singleton:

$$c'_w \in D \cup P$$

where D is the set of dictionary terms and P is defined as a set of static user-defined strings:

$$P = \{p_b \mid 1 \leq b \leq x\}$$

Note that P may include strings defined as a consequence of actions (see Section 3.3.4.3).

The set of variable mappings, C'', from name to value (i.e. variable evaluation), are in the set C'', using a temporal function C_F that evaluates text in the presence of variable names and variable matching criteria:

$$C_F(T \mid C, C') \rightarrow C'' \tag{3.5}$$

with each individual evaluation c''_t defined such that $c''_t \in S \cup P$
S is the set of synonyms defined in Equation 3.2.

In implementation, simple variable evaluation in rule conclusions (post-inference) is forced via the notational syntax (for example):

<LIT>@c_w</LIT>

where c_w is a user-defined variable name defined in Equation 3.4.

3.3.4.2 System Context Variables

Implementation practicalities dictate that the developed system adopts a closed-world approach – this means the domain expert must pre-define any assumed factual world knowledge that is required. The approach is taken to simply define meta-rules again in the form of *system* context variables with default values that can be overridden by more specific context. For example, consider the question, "*What year is it?*". The expert can define via the C-MCRDR CA system's GUI, a default value such as (`@SYSTEMyear=2020`). A standard context variable, (`@year`) will override this value if an utterance contained the appropriate variable-matching criteria (for example, "*The year is 2021*"). No reference to *year* via its matching criteria means the default value will be set to 2020 in this example. This default context has extremely useful consequences when binding queries as detailed in Section 3.3.7.

Table 3.9: System context variable example

Variable	Matching Criteria
@year	/RE:the year is (\d)(\d)(\d)(\d)$
@SYSTEMyear	set(2020)

3.3.4.2.1 System Context Variable Definition

C^s is a set of size d of system variable IDs:

$$C^s = \{c_g^s \mid 1 \leq g \leq d\}$$

Each c_g^s has an associated value:

$$c_g'^s \in P^s \tag{3.6}$$

with P^s defined as set of static strings:

$$P^s = \{p_a^s \mid 1 \leq a \leq q\}$$

If input contains c_t' then c_t overrides c_g^s iff $c_g^s =$ "$@SYSTEM$" $+ c_t$

3.3.4.3 Context Variable Actions

The presence of a context variable value may trigger the one-time assignment of values to one or more other variables. This allows significant modification of system-retained generated context. For example, consider an animal taxonomy domain. Assume the current animal being discussed is a frog. Other context variables such as `species`, `class`, `genus` can be pre-defined and made the target of actions applied to the variable `animal` if its value matches "frog".

An action is triggered (and evaluated during inference, see Section 3.3.5) by the existence of a value for a specified context variable (c_t). Action consequences (the trigger target context variable's value) can either be static values, or the value of other existing context variables.

3.3.4.3.1 Context Variable Action Definition

An action is triggered by the presence of a value c_t'' of a variable c_t to instantiate variable c_{t2}:

$$c_{t2}'' = p_t \mid \exists\, c_t'' \in S \cup P$$
$$p_t \in P$$
$$c_t \in C$$

Or alternatively:

$$c''_{t2} = c''_{t3} \mid \exists c''_t \in S \cup P$$
$$c''_{t3} \in S \cup P$$
$$c_t \in C$$

Evaluating c_{t2} gives a static string (p_t), or the value of another variable (c_{t3}).

The action trigger can also be limited by the evaluation of source c_t (i.e. its value, c''_t) containing a substring (defined by the operator $\subseteq_s$) of another c''_{t4}:

$$c''_{t2} = p_t \mid \exists\ c''_t \in S \cup P$$
$$p_t \in P$$
$$c_t \in C$$
$$c''_{t4} \subseteq_s c''_t$$
$$c''_{t4} \in S \cup P$$

Or alternatively:

$$c''_{t2} = c''_{t3} \mid \exists\ c''_t \in S \cup P$$
$$c''_{t3} \in S \cup P$$
$$c_t \in C$$
$$c''_{t4} \subseteq_s c''_t$$
$$c''_{t4} \in S \cup P$$

Evaluation of c_{t2} is defined as either a static string, p_t or as the value of another context variable, c_{t3}, provided the context variable c_t is defined and has a value that contains the value of the context variable c_{t4} as a substring.

3.3.5 C-MCRDR Inference

Case dialog text T undergoes transformation to become T' prior to inference. As part of this transformation, any synonym substrings, $s_i \in S$, in the input case are replaced by their unique, associated dictionary terms, $d_i \in D$ from dictionary replacement (Section 3.3.2.1). Relevant

context (i.e. original synonym text) is stored by pre-defined context variables, $c_t \in C \mid c''_t = s_i$ (Section 3.3.4.1) for use post-inference. The fully transformed input text, T' is then passed to the C-MCRDR knowledge-base for inference according to the algorithm defined in Table 3.3.

The inference request will result in a *set* of inference results. The set of rules whose antecedents are satisfied by the current case is defined as the set I. I contains satisfied rules r_i where each r_i has no satisfied children (as defined by standard MCRDR inference):

$$I = \{r_i \mid 1 \leq i \leq z\} \mid satisfied(r_i), \quad not(satisfied(child(r_i))), \quad z \leq |KB| \tag{3.7}$$

Note z is bound by the total number of rules in the knowledge-base. A rule conclusion, r'_i is the classification of the satisfied rule r_i, which is produced from the function R_C i.e.

$$R_C(r_i) \rightarrow r'_i \; \forall r_i \in I \tag{3.8}$$

If $|I| = 0$, a default response (r'_0) is returned (corresponding to rule zero, r_0).

The inference function, I_F, is defined to apply to case text T' and with the presence of context variable evaluations in the set C'' to produce the set defined in Equation 3.7 i.e.

$$I_F(T' \mid C'') \rightarrow I \tag{3.9}$$

Then the set of rule conclusions, I' can be defined via:

$$R_C(I_F(T' \mid C'')) \rightarrow I' \tag{3.10}$$

(*assuming R_C can be applied to a set of rules to produce a set of conclusions*)

Equations 3.9 and 3.10 show inference is affected by both the actual case text as well as current context, which is one of the primary differences between C-MCRDR and MCRDR.

The final C-MCRDR inference output (but prior to post-inference processing), is defined as O_t, which is the multiple string concatenation ($\sum$) of each individual rule consequence, r'_i corresponding its counterpart rule, $r_i \mid r_i \in I$:

$$O_t = \sum_{n=1}^{|I|} R_C(r_n) \tag{3.11}$$

Or alternatively,

$$O_t = \sum_{n=1}^{|I'|} r'_n, \; r'_n \in R_C(I_F(T' \mid C'')) \tag{3.12}$$

3.3.6 Post-Inference

The results of inference in C-MCRDR in the C-MCRDR CA system are classifications of a user utterance. These classifications may include references to queries or variable values (called literals in the implementation - see Tables A.8 and A.11 in Appendix A) that must be resolved (bound) before a final response is returned to the user. Inclusion of unresolved classifications can significantly reduce the number of rules and rule redundancy in the knowledge-base – final classification specificity is achieved by binding conversational contextual data to the inference results in a post-inference phase by the system's output parser.

3.3.6.1 Database Querying

NLIDB interaction is supported when the specific post-bind classifications results include references to database queries. The developed system supports the domain expert to create query templates by a suitable GUI interface, and it guides the expert to chose query-binding context (typically by variable evaluation), and the query template results are XML-based (for RDBMS independence). The query templates abstract the underlying referential table schema by a simple one-to-one mapping, so queries are potentially reusable in other domains that have compatible table schema. Standard queries (such as a simple primary-key join) can be defined for cross-domain applicability by the system GUI (see Figure 3.3).

Table 3.10 shows example XML generated by the developed system based on the interaction with the GUI shown in Figure 3.3. A further abstraction of schema-mapping is possible, for instance a lookup-table mapping abstract table and field descriptors to actual values, however this was not implemented in the C-MCRDR CA system. The output parser in the implementation binds variable references to values (as described in Equations 3.5 and 3.6 from Sections 3.3.4 and 3.3.4.2 respectively) and it produces RDBMS-specific SQL queries such as that shown in Table 3.11 ("2020" is the parsed value of the context variable `@SYSTEMyear` which is then partially matched in SQL by the "%" operator). The full query XML specification is included in the Appendix section (see Tables A.6 and A.7), but it should be noted the specification does not provide complete coverage of query semantics – only inner joins are supported with the assumption all referenced tables are normalised and in 3rd normal form (3NF).

Results of a query are injected into the conclusion text by a simple query-reference tag-replacement, in this example (corresponding to Table 3.10), placeholders matching `<QREF>48</QREF>` will be replaced by the query result text.

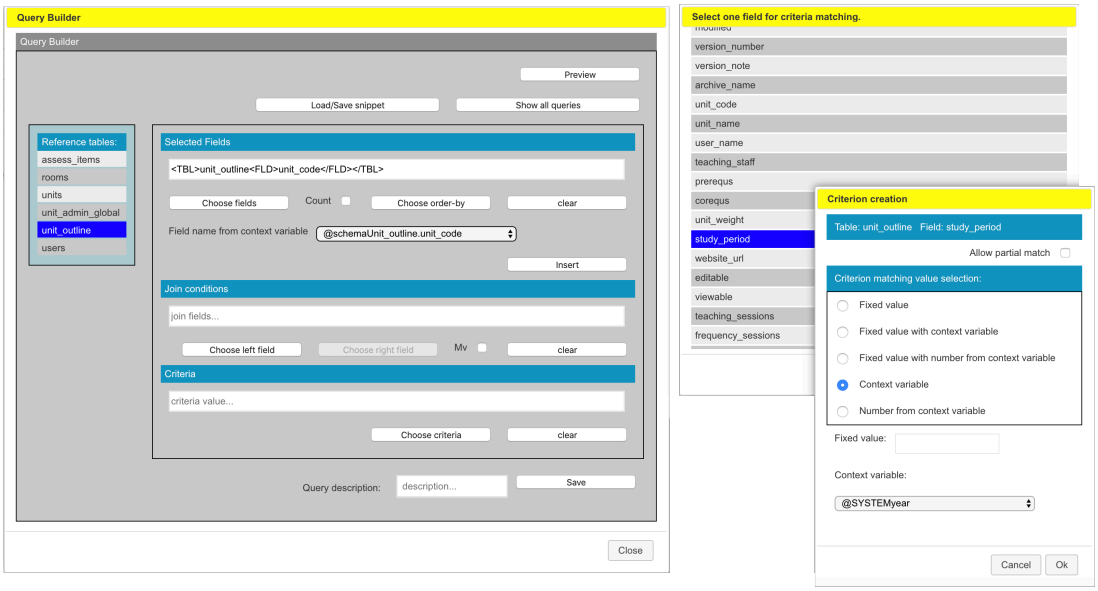


Figure 3.3: Database query builder interface

Table 3.10: Example query XML

Query XML
<QUERY>48 <TBL>unit_outline <FLD>unit_code</FLD> </TBL> <CRT>unit_outline <FLD>study_period</FLD> <CCX>@SYSTEMyear<PAR>Y</PAR></CCX> </CRT> </QUERY>

Table 3.11: Example query SQL

Query SQL

`SELECT unit_outline.unit_code`
`FROM unit_outline`
`WHERE unit_outline.study_period LIKE "%2020%"`

3.3.6.1.1 Query Reference Replacement Definition

Assuming post-inference textual results, O_t as defined in Equation 3.12, Q is defined as a set of predefined references to stored queries, $|Q| = Q_N$:

$$Q = \{q_n \mid 1 \leq n \leq Q_N\}$$

O_t is transformed to O'_t such that each $q_n \in Q$ reference is replaced by its associated database query evaluation, q'_n by the query evaluator function, Q_E:

$$Q_e(O_t) \rightarrow O'_t \mid \forall q_n \in Q_t : q_n \rightarrow q'_n \qquad (3.13)$$
$$q'_n \in P^q$$

P^q is the set of all possible static string responses from the database's RDBMS over queries from Q_t.

If an evaluated query, q'_n, is the empty string ($\varnothing_s$), a replacement function can be used - the semantics of this replacement are described in Section 3.3.6.1.3.

3.3.6.1.2 Query Reference Syntax

XML-style syntax (see Appendix A, Table A.8) is used to reference defined queries – the developed system GUI provides automatic reference insertion (precluding the domain expert from being required to learn syntax), however the referential syntax can be manually typed if required.

Each q_n is defined:

$$q_n = \texttt{<QREF>}n\texttt{<DESC>}s\texttt{</DESC></QREF>} \mid 1 \leq n \leq Q_N, \qquad (3.14)$$
$$s \in A^s$$

A^s is a set of any arbitrary strings (including the empty string, $\varnothing_s$). s is simply a descriptive placeholder used to describe the query reference prior to post-inference processing.

3.3.6.1.3 Query Result Replacement Operator

Query results that are present in O'_t can be overridden during post-processing, which are indicated by the wrapped presence of a conditional text-replacement operator. Each rule consequence (r'_i) present in O_t from Equation 3.12 may contain zero or more query reference replacements terms (i.e pre-evaluation query references, q_i, surrounded by replacement syntax). Each such query reference replacement term, t_{ij} within an individual r'_i uses a replacement

token syntax shown in Equation 3.16. Consider the partial (substring) conclusion terms, t_{ij} defined by:

$$t_{ij} \subseteq_s r'_i \mid 1 \leq j \leq n \tag{3.15}$$
$$\subseteq_s \textit{ is defined as the substring operator}$$

The t_{ij} term is comprised of the *query text replacement operator* syntax, defined as the `REPLACE` operator, R:

$$t_{ij} = \texttt{<REPLACE>}q_n\texttt{<REPLACE-EMPTY>}s_e\texttt{<REPLACE-OVERRIDE>}s_o\texttt{</REPLACE>} \tag{3.16}$$
$$s_e, s_o \in A^s$$

Application of the query evaluator function, Q_E gives the term t'_{ij}:

$$Q_E(t_{ij}) \rightarrow t'_{ij} \ \forall q_i \subseteq_s t_{ij} \tag{3.17}$$

In other words,

$$t'_{ij} = \texttt{<REPLACE>}q'_n\texttt{<REPLACE-EMPTY>}s_e\texttt{<REPLACE-OVERRIDE>}s_o\texttt{</REPLACE>} \tag{3.18}$$

(recall q'_n is the result of evaluating query reference q_n)

The semantics (evaluation) of the `REPLACE` operator, R then are:

$$R(t'_{ij}) \rightarrow t''_{ij}$$
$$R(t'_{ij}) = \begin{cases} q'_n \mid q'_n \neq \varnothing_s, s_o = \varnothing_s \\ s_e \mid q'_n = \varnothing_s \\ s_o \mid q'_n \neq \varnothing_s, s_o \neq \varnothing_s \end{cases} \tag{3.19}$$
$$(\varnothing_s \textit{ represents the empty string})$$

In other words, if the result of query reference q_n evaluation is an empty string ($\varnothing_s$), a default response (s_e) can be provided. Alternatively, if evaluation returns a non-null query result, the actual result can be discarded and an alternative response can be returned (s_o). A simple example (Table 3.12) shows the usefulness of the semantics of Equation 3.19. If the result of $query_1$ is empty, then the final, post-processed inference response (O''_t) to the user input *Is there an exam* would be *There is no exam!* Alternatively, if an exam duration is found, the response is *Yes!* instead of the actual duration value.

Table 3.12: Query replacement example

Attribute	**Value**
user dialog	*Is there an exam?*
$query_1$	*SQL code for getting exam duration from relevant table*
q_1	`<QREF> 1 <DESC> exam existence </DESC></QREF>`
$Rule_1$ (r_1) condition	`dialog CONTAINS(`*Is there an exam?*`)`
$Rule_1$ conclusion (r'_1)	t_1
t_1	`<REPLACE>`q_1 `<REPLACE-EMPTY> There is no exam!` (s_e) `<REPLACE-OVERRIDE> Yes!` (s_o) `</REPLACE>`
O_t	t_1
q'_1 ($Q_E(q_1) \rightarrow q'_1$)	180 minutes

t'_1	`<REPLACE>180 minutes`
	`<REPLACE-EMPTY> There is no exam!`
	`<REPLACE-OVERRIDE> Yes!`
	`</REPLACE>`

t''_1	*Yes!*
O''_t	*Yes!*
q'_1	$\varnothing_s$

t'_1	`<REPLACE>`$\varnothing_s$
	`<REPLACE-EMPTY> There is no exam!`
	`<REPLACE-OVERRIDE> Yes!`
	`</REPLACE>`

t''_1	*There is no exam!*
O''_t	*There is no exam!*

3.3.7 Rule-count Reduction

Substantial rule-count reduction in the knowledge-base is a significant beneficial side-effect to the inclusion of post-inference queries; standard MCRDR tends to create larger decision trees compared to other inductive methods (due to over-generalisation and subsequent rule refinement by the domain expert for special cases as they arise (Kang, 1995)), and the complexity of the decision tree is increased as a consequence. In domains where considerable knowledge is already present in the form of *in situ* databases (as is the case for this phase's target pedagogical domain), the potential rule-bloat can be drastically reduced by incorporating querying into the interim inference results as detailed in the previous section.

3.3.7.1 Best-case rule-count reduction

To determine the best-case reduction of rule-count in C-MCRDR compared to MCRDR, consider a perfectly balanced MCRDR decision tree that has an arbitrary number of exception rules (but a constant, consistent value for each level of the tree) per node. If the tree is of height L, then the number of nodes at level i is defined in Equation 3.20:

$$N_i = \prod_{j=1}^{i} d_j \;\Big|\; 1 \leq i \leq L \tag{3.20}$$

where each d_j is the degree of branching at level j

The single root node, corresponding to rule R_0, is defined as level or tree height 1, so $N_1 = 1$. The total number of nodes, N, in the tree is then simply the sum of the number of nodes at each level i:

$$N = \sum_{i=1}^{L} N_i \tag{3.21}$$

Equation 3.21 assumes a per-level degree of rule refinement per node, and so it is an upper-bound of the number of rules in the tree - an example decision tree to illustrate this is shown in Figure 3.4. Real decision trees will of course vary in the number of rule refinements per node.

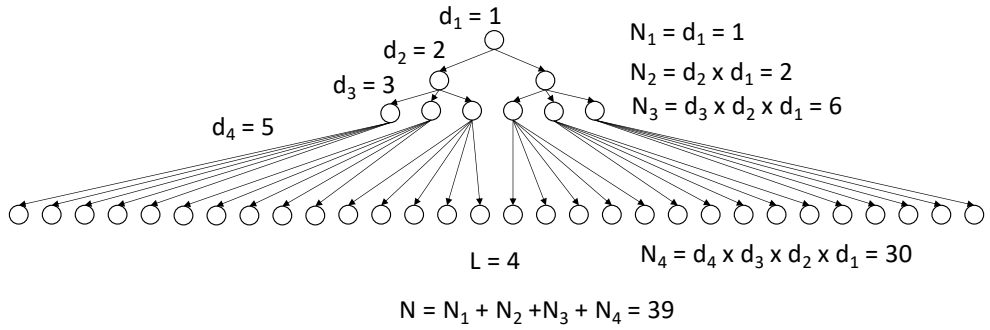


Figure 3.4: Example MCRDR decision tree node count

In terms of rule-count reduction as a consequence of using C-MCRDR with context and queries, the best-case count of nodes at each level for the C-MCRDR tree (defined as N'_i) can be determined. The C-MCRDR rule/node count is defined in an analogous manner to Equation 3.21:

$$N' = \sum_{i=1}^{L} N'_i \tag{3.22}$$

The correlation between the two counts, N and N', is determined by introducing a set of affinity factors for each level of the MCRDR tree:

$$A_i = \{\alpha_{ij} \ \ |1 \leq i \leq L\} \qquad (3.23)$$
$$1 \leq j \leq N_i$$

Each affinity set $\alpha_{ij} \in A_i$ are themselves non-intersecting sets of rules (r_k) at tree level i in the original MCRDR decision tree that can be replaced by a single summarisation rule that uses queries and context and note that there are at most N_i elements in A_i (when each $\alpha_i j$ only contains a single rule). Alternatively, the maximum size of an α_{ij} may be N_i (meaning all rules at level i are contained in the single set α_{i1}). Assuming $L(x)$ gives the tree height of node x, an affinity set of rules is defined by:

$$\alpha_{ij} = \{r_k \mid L(r_k) = i\} \qquad (3.24)$$
$$k \leq |KB|$$
$$|\alpha_{ij}| \leq N_i$$

All affinity rule sets within A_i are non-intersecting:

$$\bigcap_{j=1}^{m} \alpha_{ij} = \varnothing \mid m \leq N_i \qquad (3.25)$$

If the size of an affinity rule set, $|\alpha_{ij}| = 1$ then the set contains one rule, meaning there is no summarisation rule (or there is only a one-to-one replacement query rule which means there is no reduction in the rule-count).

Noting that this initial, defining MCRDR tree is balanced, so the degree of level i, d_i, is indicative of the number of nodes (per parent node) at that level, the best rule-reduction case is defined as where $|\alpha_{ij}| = N_i$, which implies there is only one set ($j = 1$) containing N_i rules at level i, so subsequently $|A_i| = 1$. This means every child rule of a node has been replaced by a summary rule.

The C-MCRDR rule count is defined for level i by replacing rules within each affinity set by a summarisation rule, so the number of elements (sets) in A_i is indicative of the resulting rule-count:

$$N'_i = |A_i| \qquad (3.26)$$

The final consideration in the determination of the MCRDR tree rule-count to the C-MCRDR equivalent, is that rules are replaced by summary rules in a top-down, iterative approach. This

implies affinity factors are manually re-evaluated at each level of the tree after the previous level has been transformed. In other words, the node-count N_i in Equations 3.23 to 3.25 will change each replacement iteration.

It should be noted the C-MCRDR rule-count reduction considerations are for comparison purposes – the methodology does not suggest an initial MCRDR decision tree is reduced either manually or algorithmically to an equivalent C-MCRDR tree; instead the methodology in this section details *what* an equivalent C-MCRDR tree would be given the specific domain environment (i.e. values for rule affinities that are used to determine summarisation rules). In a real environment a C-MCRDR tree is constructed via knowledge acquisition in parallel to the construction of domain-specific queries given knowledge already present in the form of *in situ* databases.

3.3.7.2 Rule-count reduction example

The significant reduction in knowledge-base rule bloat is best demonstrated by an example. Figure 3.5 gives an example of a question's classification of regular polygons and instances of their colour prior to any optimisations. Ten non-root MCRDR rules are required to provide complete coverage of answering questions associated with the polygon's name and colour such as:

- "*what is the polygon with 5 sides?*" (e.g. R_3)
- "*what is the colour of the polygon with 5 sides?*" (e.g. R_8)

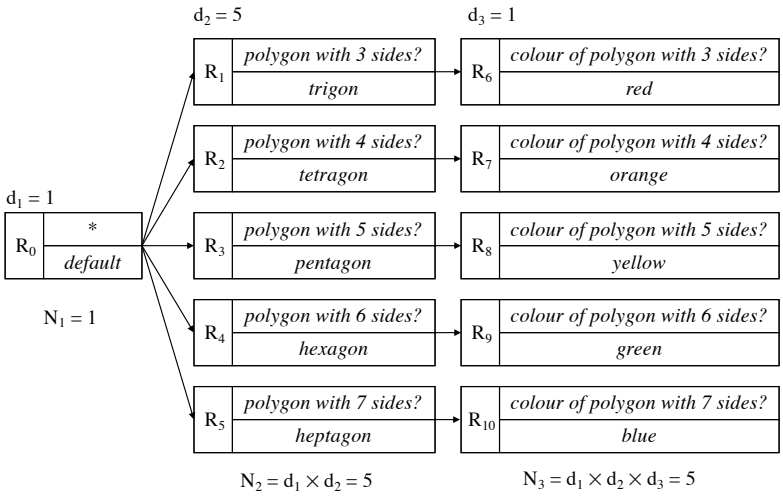


Figure 3.5: MCRDR rules prior to context and queries

The actual implementation of (for example, Rules 1 and 6) may be:

- R_1: `contains(polygon) and contains(3)` $\rightarrow$ `trigon`
- R_6: `contains(polygon) and contains(3) and contains(colour)` $\rightarrow$ `red`

Note here that R_6 is an MCRDR refinement/reclassification of the conclusion of $R1$ i.e. an EXCEPT rule in MCRDR terminology. Each rule-pair (R_1, R_6), (R_2, R_7), (R_3, R_8) etcetera is required for complete coverage of all polygon shapes and associated colours. This does not scale well in terms of rule-count when the number of instances of polygons increases.

Table 3.13: Pre-existing domain database "polygon"

names		***colours***	
sides	***name***	***type***	***colour***
3	trigon	trigon	red
4	tetragon	tetragon	orange
5	pentagon	pentagon	yellow
6	hexagon	hexagon	green
7	heptagon	heptagon	blue

Assuming the existence of a domain database (Table 3.13), the knowledge-base can be reduced significantly – this is shown in Figures 3.6, 3.7 and 3.8. The knowledge-base had been reduced to two C-MCRDR rules in addition to the default rule (by manual rule inspection and replacement) using the definitions from Section 3.3.7.1, the introduction of summarisation queries and the continuation of topical context.

Topical context (via inference result stacking) is evident here as rule R_6's antecedent need only contain for example:

- R_6: `contains(colour)` $\rightarrow$ `<QREF>2</QREF>`

  instead of

- R_3: `contains(polygon) and contains(3) and contains(colour)` $\rightarrow$ `<QREF>2</QREF>`

In addition, the literal C-MCRDR rule conclusions containing query references are shown – these are parsed by the system's output parser and the corresponding query XML code (as shown in Table 3.17) is executed with the relevant binding context (the `@sides` variable, which is assigned the numerical portion of the `/sides/` term as defined in Tables 3.14 and 3.15).

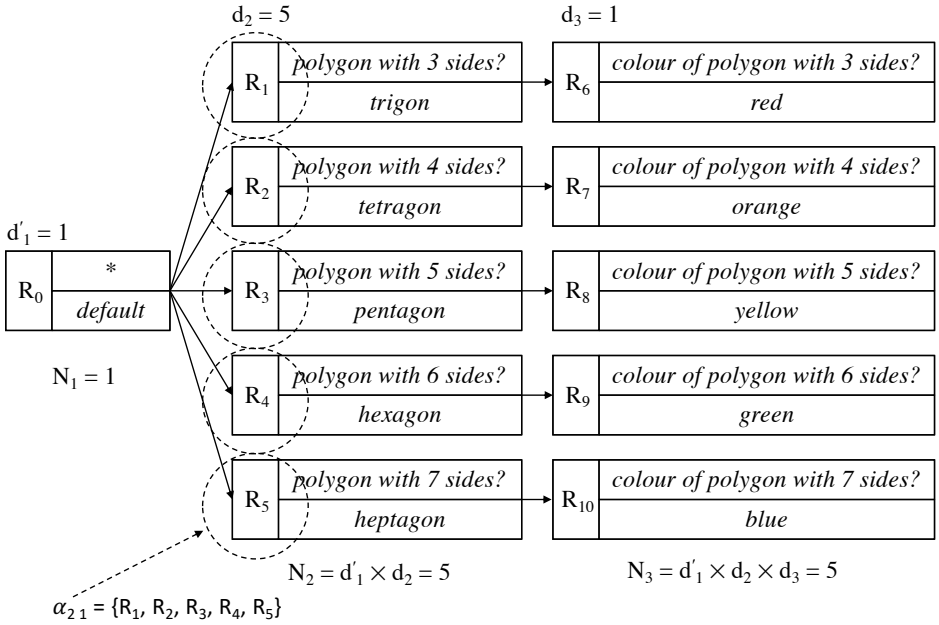


Figure 3.6: MCRDR grouping of rules at level 2

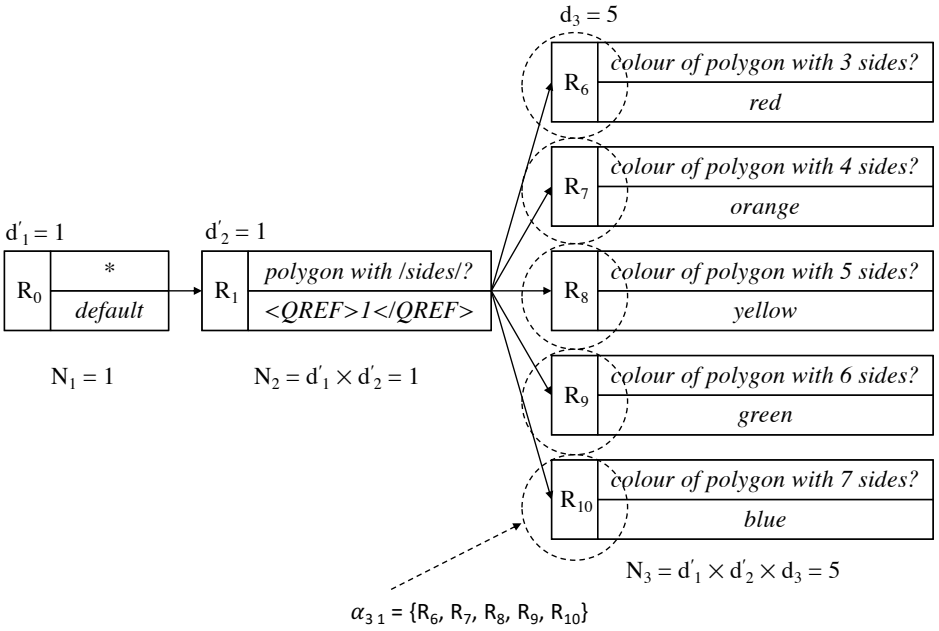


Figure 3.7: MCRDR grouping of rules at level 3

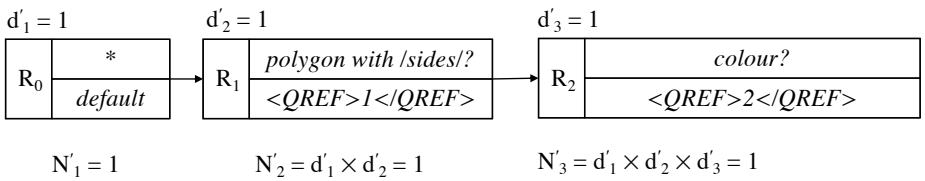


Figure 3.8: C-MCRDR rules after addition of context and queries

Table 3.14: Dictionary term definition

Term	**Synonym**	**Input matches**
/sides/	`/RE:`(\d) *sides*	*7 sides* *99 sides*

Table 3.15: Context variable definition

Variable	**Criteria**	**Example Evaluation**
@sides	`num(`*/sides/*`)`	*7* *99*

Query 1 (<`QREF`>`1`) in Figure 3.8 is a simple select (*name*) on the *names* table with *sides* as the key, whereas Query 2 selects *colour* based on a table join matching *name* with *type*.

The example, final knowledge-base shown in Figure 3.8 now allows C-MCRDR conversations as shown in the example, Figure 3.9.

> "*what is the name of the polygon with 7 sides?*" → "*heptagon*".
> "*what is its colour?*" → "*blue*"

Figure 3.9: Example C-MCRDR conversation

This example exhibits an 72.7% reduction of the rule count (11 rules to 3) in the KB compared to the non-query example shown in Figure 3.5.

If more general, but highly-consistent MCRDR trees are considered that are perfectly balanced, and have constant, global degrees, then the percentage rule-count reduction can be calculated for a variety of affinity set sizes (see Figure 3.10). For simplicity the assumption is made that the affinity set size ($|\alpha|$) is constant for every level of the tree (which would not likely occur in real domains). For example, an MCRDR tree with degree 5 exhibits an 80% reduction of rule-count for tree heights of 4 or more, when affinity sets of rules are set at 5 (meaning all child rules per node are replaced by one rule). This is of course a highly idealised example that sets an upper-bound on the potential rule reduction count. Table 3.18 gives a summary of the before (n) and after (n') tree rule-counts (when the affinity set size matches the tree's degree) for a given height (level, L). Note the node count for balanced trees of degree n is:

$$N = \frac{n^L - 1}{L - 1}$$

Table 3.16: Example query 1 and 2 SQL

Query 1 SQL	**Query 2 SQL**
SELECT name FROM names WHERE sides LIKE “@sides”	SELECT colour FROM names, colours WHERE names.name = colours.type AND sides LIKE “@sides”

Table 3.17: Example query 1 and 2 XML

Query 1 XML
<QUERY>1 <TBL>names <FLD>name</FLD> </TBL> <CRT>name <FLD>sides</FLD> <CCX>@sides<PAR>N</PAR></CCX> </CRT> </QUERY>
Query 2 XML
<QUERY>2 <TBL>colours <FLD>colour</FLD> </TBL> <JOI> <JVA>names<FLD>name</FLD> <JVA>colours<FLD>type</FLD> </JOI> <CRT>name <FLD>sides</FLD> <CCX>@sides<PAR>N</PAR></CCX> </CRT> </QUERY>

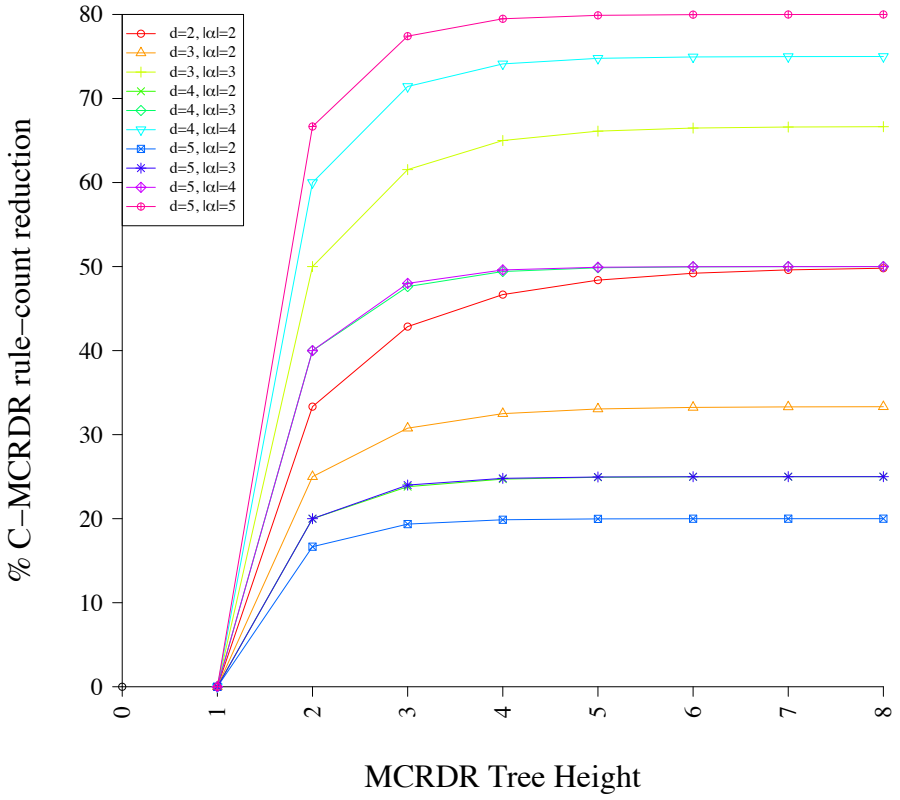


Figure 3.10: Percentage reduction of rule-count C-MCRDR

Table 3.18: MCRDR/C-MCRDR ideal rule-count comparisons

	Tree Degree ($\|\alpha\| = degree$)							
Level	2	2′	3	3′	4	4′	5	5′
1	1	1	1	1	1	1	1	1
2	3	2	4	2	5	2	6	2
3	7	4	13	5	21	6	31	7
4	15	8	40	14	85	22	156	32
5	31	16	121	41	341	86	781	157
6	63	32	364	122	1365	342	3906	782
7	127	64	1093	365	5461	1366	19531	3907
8	255	128	3280	1094	21845	5462	97656	19532

3.3.8 Brittleness Mitigation

As discussed in Section 2.2.2.3, a paraphrasal approach (Oh et al., 2015; Madnani and Dorr, 2010; Huang et al., 2019; Yang et al., 2019; Hunt et al., 2019) which could be considered a localised, limited ontological method is adopted by this thesis research. The ontological dictionary is manually populated by the domain expert to provide sufficient linguistic variation for a paraphrased term which helps mitigate brittleness. The user is also prompted with examples of utterances that are understood from the current context – this is triggered if the last user utterance only satisfied the default root rule, although a meta-rule, *What can I say?* is also a trigger. In this manner, if a user is unsure of what linguistic and domain coverage the system has, they can be given hints (see Figure 3.11).

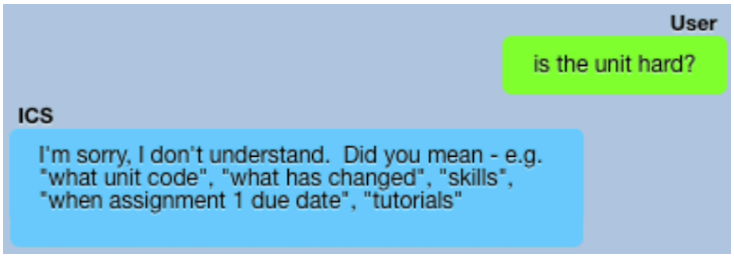


Figure 3.11: Example lookahead prompting interface

The suggested brittleness mitigation hints are selected based on attributes that will satisfy a subset of child rules (a lookahead) from the starting ruleset in the top stack frame and they are varied by randomly selecting instances in any lexical terms referred to by the rule attributes. If all child rules where hinted, the interface would become overwhelming for the user; if the same hint is always given with no linguistic (paraphrasal) variation, it may lead to boredom or "interface fatigue" for the user. For example, in the evaluated pedagogical domain, a conversational snippet might be (the underlined lexical term has been replaced between each suggestion):

User: *I like fish*

System: *I don't understand. You could try asking me "who is the lecturer?" or "what is the prior learning"?*

User: *huh?*

System: *I don't understand. You could try asking me "who is the teacher?" or "what is the prerequisite?"*

Evaluation of the paraphrasal hint brittleness mitigation is detailed in Section 4.2.6.

The developed system in the targeted pedagogical domain includes a final, domain-specific brittleness mitigation feature. If the system is unable to provide results to the client user's satisfaction, the client browser interface facilitates requesting a unit coordinator's assistance if the client enters their personal email address (Figure 3.12). This triggers an automatic inference request for the lecturer details associated with the unit currently being discussed (e.g. *what is the lecturer's email address?*), and the lecturer is then emailed a transcript of the entire conversation. The lecturer can then respond to the client's unfulfilled queries via normal email exchange.

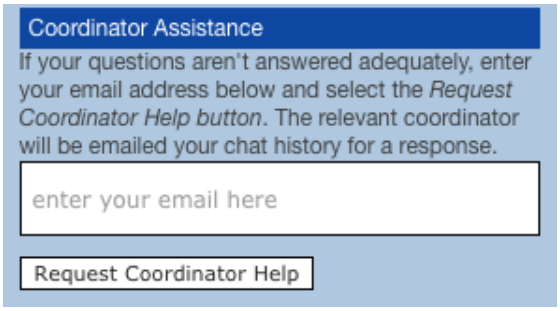


Figure 3.12: Request coordinator help interface

3.3.8.1 Meta-terms

The final post-inference, post-query replacement output, O_t'' from Section 3.3.10 is parsed to determine the presence of defined meta-term tags, `<METAHELP>`,`<METANOHELP>` and `<METAWHERE>`. These tags are used by the suggestion system to prompt the end user with valid potential input that would satisfy child rules from the current conversational context. Assume these meta-tag strings are defined as M_1,M_2 and M_3 respectively, and only one of M_1,M_2,M_3 can be a substring of O_t'' i.e.

$$M_i \subseteq_s O_t'' \mid 1 \leq i \leq 3 \;\rightarrow\; M_j \subsetneq_s O_t'' \mid 1 \leq j \leq, i \neq j$$

3.3.8.1.1 METAHELP

If $M_1 \subseteq_s O_t''$, the term in O_t'' is replaced by a string representation (a comma-separated list) of all the **child** rule antecedent conditions for each of the satisfied rules in the last non-zero size successful inference request that does not include meta tags. *Success* also implies the set of rules is non-empty, and does not just contain rule zero (R_0).

Assume the function R_A gives the sets of antecedents for rules r_i in I. $children(I)$ is defined to determine all child rules of rules in the set I such that:

(Recall from Equation 3.7):

$$\begin{aligned} I = \{r_i \mid 1 \leq i \leq z\} \mid & \, satisfied(r_i) \\ & \, not(satisfied(child(r_i))), \\ & \, z \leq |KB| \\ children(I) \rightarrow & \, J = \{r_j \mid i \leq j \leq z\}, \\ & \, parent(r_j) \in I \; \forall r_j \in J, \\ & \, child(r_i) \in J \; \forall r_i \in I \end{aligned} \tag{3.27}$$

In other words, J is the set of rules whose immediate parent rules are rules in set I.

Dictionary term replacement - each $d_i \in D$ dictionary terms found in each $R_A(r_j)$ such that $d_i \subseteq_s R_A(r_j)$, is replaced by a random synonym $s_{ij} \in S_i$ (from Equation 3.2) to improve the verbosity of the suggestion. For example, the dictionary term `/lecturers/` might be replaced by the synonym `teaching staff`. The inverse dictionary function, D'_F is defined to produce all synonyms associated with a dictionary term, analogous in reverse to Equation 3.3 where synonyms are mapped to dictionary terms:

$$D'_F(d_i) \rightarrow S_i \tag{3.28}$$

$RAND(X)$ will produce one, and only one random element from the set X:

$$RAND(S_i) \rightarrow s_{ij} \tag{3.29}$$

So for every dictionary term found in a rule's set of antecedents (given by R_A), terms are replaced by a random (but associated) synonym by the term replacement function, R_T:

$$R_T(R_A(r_j)) : d_i \rightarrow RAND(D'_F(d_i)) \; \forall d_i \subseteq_s R_A(r_j) \tag{3.30}$$

Assuming S is the current stack of inference results (as defined from the C-MCRDR inference algorithm in Table 3.3), the semantics of `METAHELP` are defined to give the result O'''_t as:

$$I = METAPOP(S) \mid M_1 \subseteq_s O''_t \tag{3.31}$$

$$O'''_t = t_b + R_T(R_A(children(I))) + t_a \mid O''_t = t_b + M_1 + t_a \tag{3.32}$$

`METAPOP(S)` in Equation 3.31 is defined to retrieve the closest, next non-empty (non rule-zero only) stack frame from the stack S that does not contain meta tags (so the current, top-of-stack frame is excluded as it includes the rule that contained the M_1 tag in its conclusion). tb (text before) and ta (text after) in Equation 3.32 are substrings surrounding the M_1 tag in the current inference output, O''_t.

METAHELP example: as an example of using the M_1 tag, assume an additional rule (R_4) has been added to the knowledge-base (see Figure 3.13), which is then satisfied by the utterance "*help*" following on from the conversation in Figure 3.9 where the last user utterance was *colour?*. Assuming the dictionary term `/area/` consists of the synonyms `area`, `size`, and `coverage` gives:

$$M_1 \subseteq_s O''_t, M_1 = \texttt{<METAHELP>}$$
$$O''_t = R_C(R_4) = R'_4 = t_b + M_1 + t_a$$
$$t_b = \text{“}\texttt{Try asking me about - }\text{”}$$
$$t_a = \text{“”}$$

As M_1 is encountered, the previous inference result stack frame rule set that excludes R_4 in the results is retrieved, thus $I = METAPOP(S) = \{R_2\}$.

$$\begin{aligned} O'''_t &= t_b + R_T(R_A(children(I))) + t_a \\ &= \text{“}\texttt{Try asking me about - }\text{”} + R_T(R_A(\{R_3\})) \\ &= \text{“}\texttt{Try asking me about - }\text{”} + R_T(\texttt{/area/}) \\ &= \text{“}\texttt{Try asking me about - }\text{”} + \text{“}\texttt{size}\text{”} \end{aligned}$$

The response would be "*Try asking me about – size.*" (assuming *size* is the random synonym chosen)

Note: R_A implicitly defaults to assume the (most common) antecedent condition is `contains` e.g. `contains(/area/)` which is then omitted in the function result (`contains(/area/)` → `/area/`). Other conditional operators (e.g. `NOT CONTAINS`, `EQUALS`) will not be omitted in order to avoid possible ambiguity.

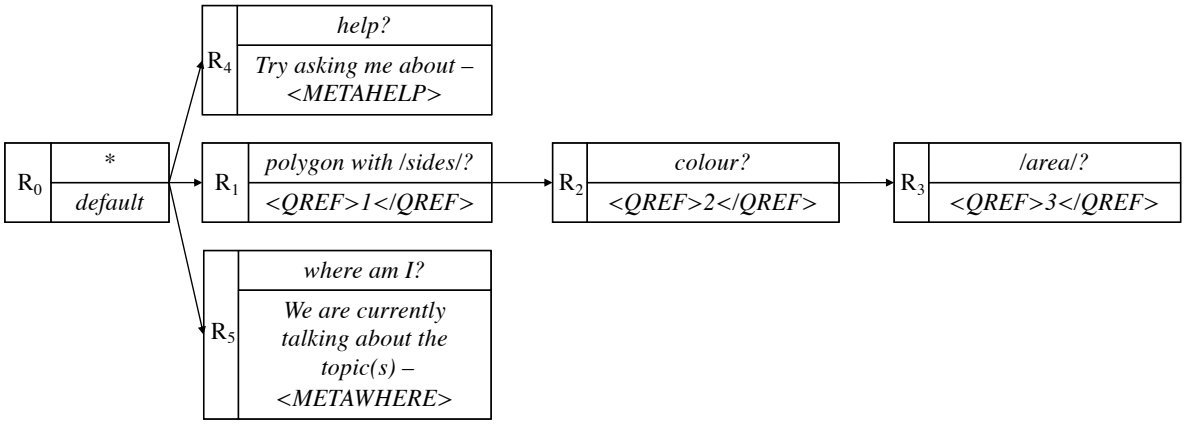


Figure 3.13: Addition of METAHELP and METAWHERE meta-rules

3.3.8.1.2 METANOHELP

An r_j is rejected (removed from the set J) if $M_2 \subseteq_s r'_j$ (recall $R_C(r_j) \rightarrow r'_j$, the conclusion of r_j). This is the sole purpose of the M_2 tag – to ensure a rule's antecedent conditions are not included in the help suggestions (some rules may not relevant as suggestions from the domain user's perspective). An example of this is where a dictionary contains a synonym which is a regular expression, which is not intelligible for a user to process. As a consequence, such dictionary terms containing non-simple synonyms are marked to include the M_2 tag to preclude them being offered as utterance hints. The actual semantics of `METANOHELP` are detailed in Equation 3.33 – a random synonym is not produced ($\varnothing_s$, the empty string) for any dictionary terms in a rule that contains `METANOHELP` in its conclusion.

$$R_T(d_i) \rightarrow \varnothing_s \mid M_2 \subseteq_s r'_j \; \forall d_i \subseteq_s R_A(r_j) \tag{3.33}$$

3.3.8.1.3 METAWHERE

Similar to `METAHELP`, the `METAWHERE` M_3 tag in an r'_i will direct the system response to the domain user to contain the **previous** successful (non R_0) inference result's satisfied rule antecedent *conditions* given by R_A. In contrast to `METAHELP`, the set of rules satisfied as being present in the set I is observed, not their children i.e. if $M_3 \subseteq_s O''_t$, the term is replaced in O''_t by a comma-separated list of all antecedent conditions of all $r_i \in I$ *for the previous inference stack frame*. The semantics of `METAWHERE` is defined to give the result O'''_t as:

$$I = METAPOP(S) \mid M_3 \subseteq_s O''_t \tag{3.34}$$

$$O'''_t = t_b + R_T(R_A(I)) + t_a \mid O''_t = t_b + M_3 + t_a \tag{3.35}$$

In essence, the presence of the tag is asking post-inference processing to detail the list of antecedents of rules satisfied in the previous successful inference request.

METAWHERE example: as an example of M_3, assume an additional rule (R_5) has been added to the knowledge-base (see Figure 3.13), which is then satisfied by the utterance "*where am I?*" following on from the conversation in Section 3.3.8.1.1. This gives:

$M_3 \subseteq_s O''_t, M_3 =$ `<METAWHERE>`

$O''_t = R_C(R_5) = R'_5 = t_b + M_3 + t_a$

$t_b =$ "`we are currently talking about the topic(s) - `"

$t_a =$ ""

As M_3 is encountered, the first previous inference result stack frame rule set that excludes any meta tags in the results is retrieved, thus $I = METAPOP(S) = \{R_2\}$.

$$\begin{aligned} O_t''' &= t_b + R_T(R_A(I)) + t_a \\ &= \text{“}\texttt{we are currently talking about the topic(s) - }\text{”} + R_T(R_A(\{R_2\})) \\ &= \text{“}\texttt{we are currently talking about the topic(s) - }\text{”} + R_T(\texttt{colour}) \\ &= \text{“}\texttt{we are currently talking about the topic(s) - }\text{”} + \text{“}\texttt{colour}\text{”} \end{aligned}$$

The response would be “*We are currently talking about the topic(s) – colour.*”

3.3.8.1.4 Meta-term Summary

Three meta-terms (M_1,M_2 and M_3) are defined; the post-inference, query replacement textual result O_t'' is parsed for the presence of any of the terms resulting in the final textual output, O_t''':

$$O_t''' = \begin{cases} O_t'' \mid M_1, M_2, M_3 \subsetneq_s O_t'' \\ t_b + R_T(R_A(children(I))) + t_a \mid M_1 \subseteq_s O_t'' \\ t_b + R_T(R_A(I)) + t_a \mid M_3 \subseteq_s O_t'' \end{cases} \tag{3.36}$$

3.3.9 Speech Considerations

The primary data attributes used for inference in this thesis are textual utterances by domain users. However, it was determined a speech interface can also enhance the natural language services provided by an implemented system. The second phase of this thesis' methodology considers using alternative speech-to-text devices, such as *Google Home* and *Amazon Echo*, and their associated ASR performance is evaluated (in terms of the correctness of the speech-to-text results). In the C-MCRDR CA phase methodology however, speech services are restricted to services offered by the Google Chrome browser as it provides embedded JavaScript support for API calls to the Google Speech API (Google, 2018*b*). For speech-to-text services, the browser activates the host system's microphone and records speech as compressed Free Lossless Audio Codec (FLAC) data format (Coalson, 2016), which is then sent for Automatic Speech Recognition (ASR) via Google's servers. Text-to-speech services similarly use Google's servers to render text as audio data (as e.g. mp3 format).

3.3.9.1 Speech To Text

Prior to Equation 3.1, where text T is replaced by associated dictionary terms, the text source can be the result of ASR speech-to-text recognition, producing text T_s:

$$ASR(\text{speech}) \rightarrow T_s \tag{3.37}$$

Original lexical or phrasal utterances in T_s are then pattern-matched against a set of correction rules (which are regular expressions constructed by a suitably constrained and simplified GUI interface in order to provide abstraction of esoteric regular expression syntax) – approximate string matching algorithms (Raghavan and Allan, 2005; Ukkonen, 1992; Hakak et al., 2019; Sidorov et al., 2014) such as the string edit distance (Levenshtein, 1966; Hall and Dowling, 1980; French et al., 1997; Zobel and Dart, 1995) could have been used here for a more generalised approach, but to maintain interface simplicity, this was not done – this may be considered in future work. The correction rules are defined as a sequence of ordered steps by the domain expert (the conversational author) as a pre-processing step applied to speech-to-text input, T_s. Corrections will be ignored when the user interface to the conversation system uses text directly. Please see Figure 3.14 for an example of the C-MCRDR CA system's javascript interface for defining global correction definitions.

RE_G is defined as an ordered set of user-defined regular expressions, e_i that have corresponding text replacement actions:

$$RE_G = \{e_i \mid 1 \leq i \leq n\} \tag{3.38}$$

n is the total number of speech-to-text corrections

t_0 is specified as the initial raw utterance (T_s) and t_n as the final post-correction utterance after all global corrections have occurred – so t_n is defined as the recursive process:

$$\begin{aligned} t_0 &= T_s \\ t_1 &= e_1(t_0) \\ t_n &= e_n(t_{n-1}) \end{aligned} \tag{3.39}$$

$e_i(t)$ is the evaluation of the text correction for regular expression e_i to text t. The final speech-to-text correction result, t_n becomes the normal input for pre-inference processing, T.

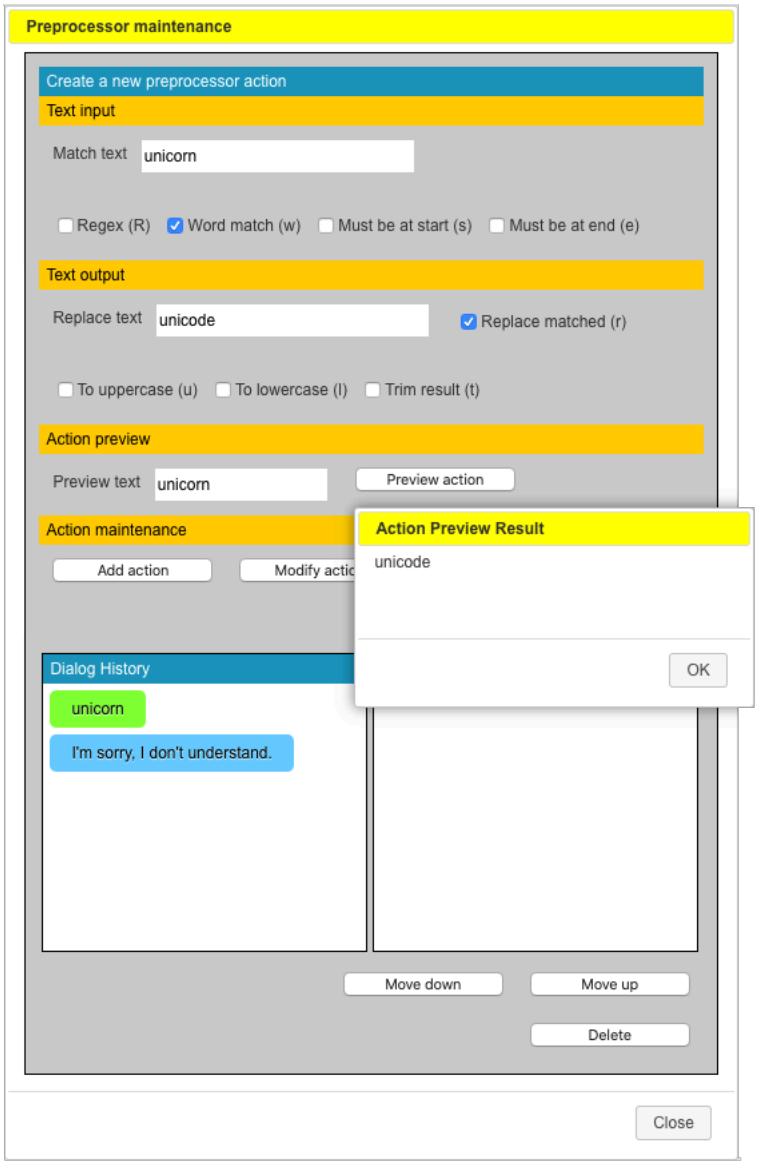


Figure 3.14: C-MCRDR CA system global definition interface

3.3.9.2 Text To Speech

The client browser JavaScript code calls Google's Speech API for text narration, text-to-speech. The domain expert can define speech-correction rules for utterances that are determined to be mis-pronounced – this typically occurs for proper nouns.

An ordered set of regular expression correction rules RE'_G, similar to those as used for speech-to-text correction, is applied to the final, post-inference, post-query, post-meta-term text, O'''_t, however each rule will normally specify a near-phonetic correction to text.

$$RE'_G = \{e'_i \mid 1 \leq i \leq m\} \tag{3.40}$$

m is the total number of text-to-speech corrections

Evaluation of each regular expression e'_i applied to O'''_t is defined by:

$$\begin{aligned} t'_0 &= O'''_t \\ t'_1 &= e'_1(t'_0) \\ t'_n &= e'_n(t'_{n-1}) \\ T_t &= t'_n \end{aligned} \tag{3.41}$$

T_t, the corrected text to be used as speech output, is sent to Google's text-to-speech (TTS) servers:

$$TTS(T_t) \rightarrow \text{speech} \tag{3.42}$$

It must be noted the original text, O'''_t is still presented as the text-based result for text-only interfaces.

3.3.10 Data Transformation Summary

The transformation of a user's input dialog utterance text (T) to the final system response text (O''_t) is shown in Table 3.19. Stage 5 (*Context variable evaluation 2*) refers to context variable evaluation post-inference (due to context variable action directives in rule conclusions).

Table 3.19: Data transformation summary

#	**Description**	**Transformation**
1	User text or speech-to-text utterance	$U, T_s \rightarrow T$
2	Context variable evaluation 1	$C_F(T \mid C, C') \rightarrow C''$
3	Dictionary term evaluation	$D_F(T) \rightarrow T'$
4	Inference	$\sum_{n=1}^{\lvert I' \rvert} r'_n,\ r'_n \in R_C(I_F(T' \mid C'')) \rightarrow O_t$
5	Context variable evaluation 2	$C_F(O_t \mid C, C') \rightarrow C''$
6	Query evaluation	$Q_E(O_t) \rightarrow O'_t$
7	Query result replacement	$R(O'_t) \rightarrow O_t''$
8	Text-to-speech	$O_t'' \rightarrow T_t$

3.3.11 C-MCRDR CA System Implementation

C-MCRDR was incorporated in the development of the C-MCRDR CA system where the implemented application was designed to be non domain-specific. The architecture for the processing of user utterance to final system response can be seen in Figure 3.15. Not shown are the knowledge acquisition (KA) components (an example of the GUI can be seen in Section 3.3.12). The KA components broadly consist of a *rule builder*, *conclusion builder*, *query builder*, *dictionary builder*, *context variable builder*, *speech pre- and post-processor builder* and an *inference viewer* (which can be used to provide an *explain* facility, a necessary KBS component (Dhaliwal, 1996)). The main development environment consists of a Java EE application (with a Tomcat Application Server back-end) and client web interfaces (for domain users as well as interfaces for the domain expert to perform knowledge acquisition) written in a combination of JSP and Javascript/JQuery.

Client (user) input arrives at the top left of Figure 3.15 and the final response occurs at the bottom left. The system module locations of intermediate transformations of the input utterance (S_p) and system response (starting with O_t) can also be seen (refer to Sections 3.3.2 to 3.3.9.2).

3.3.11.1 Architectural Component Descriptions

The main purpose of each architectural component stage shown in Figure 3.15 are as follows:

- *Client Browser* (input) – the user's utterance (direct text (U) or Speech to Text (T_s));
- *STT Preprocessing* (Section 3.4.2.1) – domain-specific corrections (Global ASR Correction) of incorrect ASR transcription (output T);
- *Context Variable Lookup* (Sections 3.3.4.1 and 3.3.4.2) – matching (and storage) of context in user's utterance to pre-defined variable criteria (output T);
- *Context Variable Actions* (Section 3.3.4.3) – pre-defined actions (targeting context variables) that are triggered by presence of other context (evaluated during inference);
- *Dictionary Lookup* (Section 3.3.2.1) – a greedy pattern-match of synonyms in the user's utterance to a representative dictionary term (rule antecedents use terms rather than the synonyms). The user's utterance is modified to replace synonyms by dictionary terms (output T');
- *C-MCRDR Inference* (Section 3.3.5) – the result of (possibly multiple) MCRDR inferred classifications of the modified user's utterance (T'). A classification (conclusion) is the raw

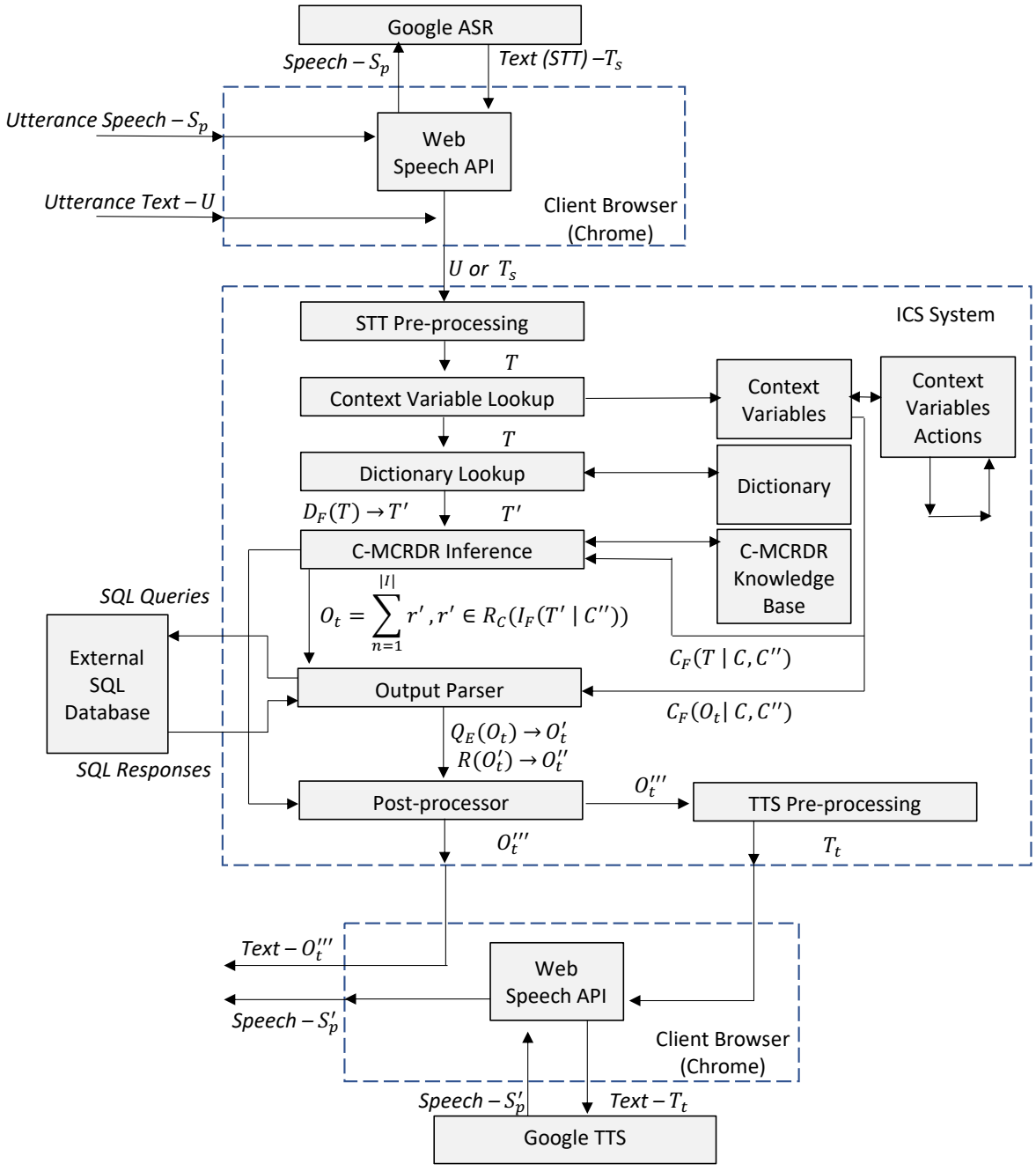


Figure 3.15: C-MCRDR CA system architecture

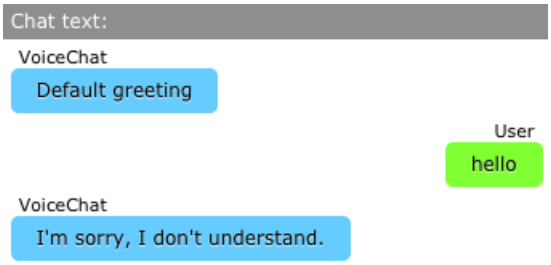
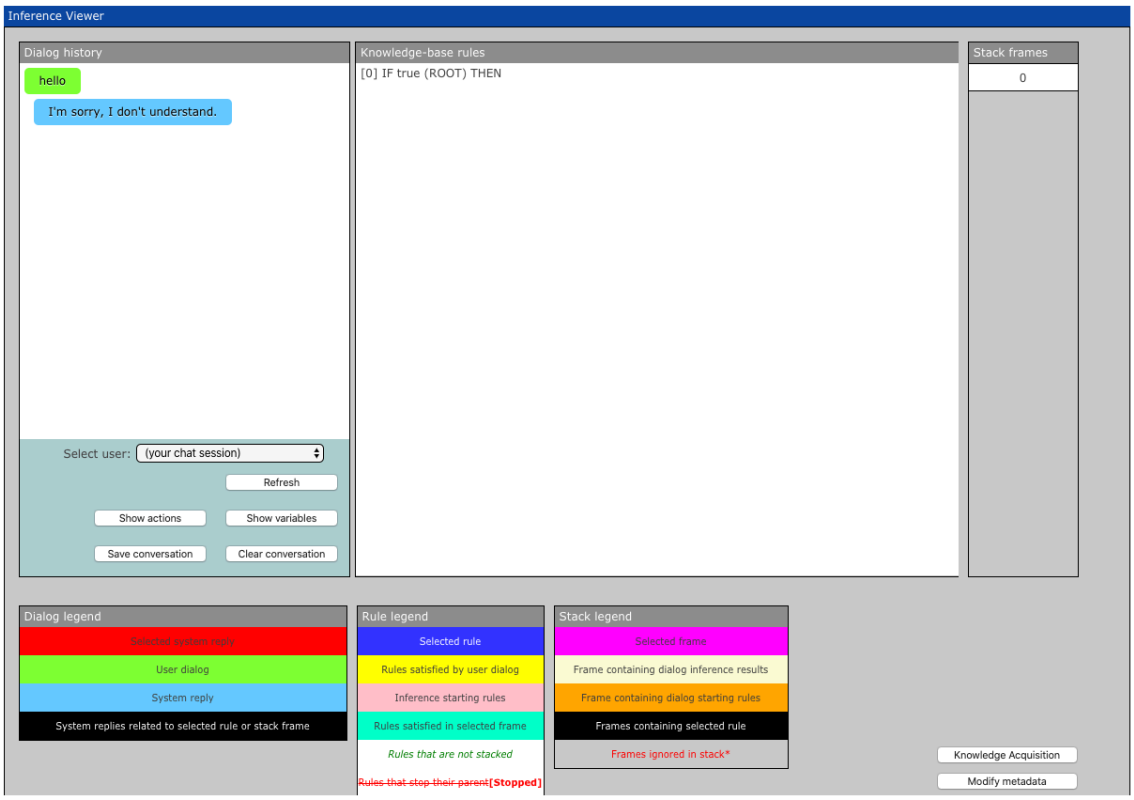
response text to be returned to the user prior to post-inference query resolution (bound by any previously found associated context) (output O_t);

- *Output Parser* (Sections 3.3.6.1.1 and 3.3.6.1.3) – context variable references found in the C-MCRDR inference response are evaluated to values and query references are replaced by the results of the associated (stored) queries being executed against the external SQL database. The query replace operator produces the final result (output O''_t);
- *Post-processor* (Section 3.3.8.1) – meta-tokens present in the C-MCRDR conclusion are removed and the response is determined by their value (output O'''_t);
- *TTS Pre-processing* (Section 3.3.9.2) – rule-based corrections (primarily for pronunciation correction) of text prior to a request to in-browser Google *TTS – Text to Speech* (output T_t);
- *Client Browser* (output) – the final text response (output O''_t) of the system to the user's utterance and a corrected, spoken equivalent of the response (output $T_t \rightarrow S'_p$).

3.3.12 Simple Example of Knowledge Acquisition Process

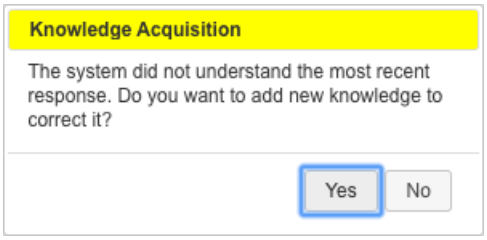
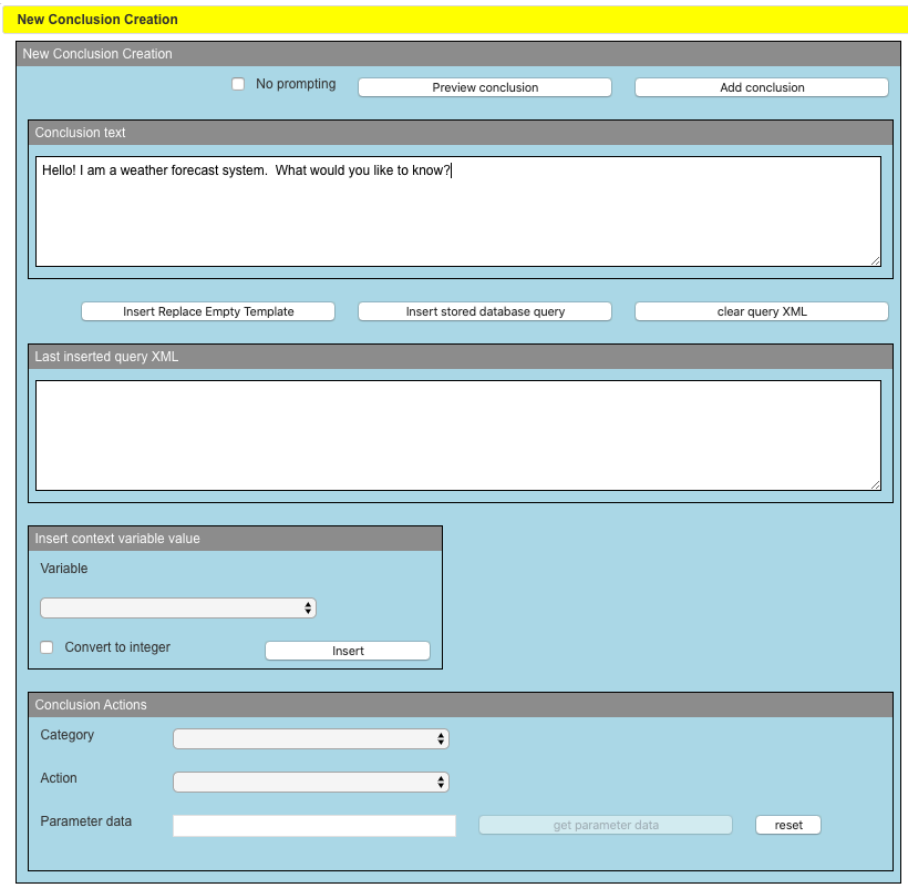
This section very briefly demonstrates the knowledge acquisition process through the C-MCRDR CA system implementation user interface (see Section 3.3.11) as shown in Table 3.20. A simple workflow is shown – the example considers a weather forecasting domain. Standarised questions such as requests for standard physical attributes like temperature, pressure, humidity, precipitation and so on could refer to an external database that contains regularly maintained and updated data for named schema items that are included in defined queries. The example shown is the beginning of the process and the conversational author is defining the first intent – a welcome to the system.

Table 3.20: Example Knowledge Acquisition Process

Step	Comment
1	The *Default greeting* initialisation response is set via an administrative interface, which would usually indicate an expected initial user utterance (such as *hello*). Here the knowledge-base contains only the default rule, and so an initial user utterance is misclassified as evidenced by the default root rule response. 
2	The conversational author initiates knowledge acquisition (through an administrative interface) in order to refine the system's response to the user's utterance. 

Continued on next page

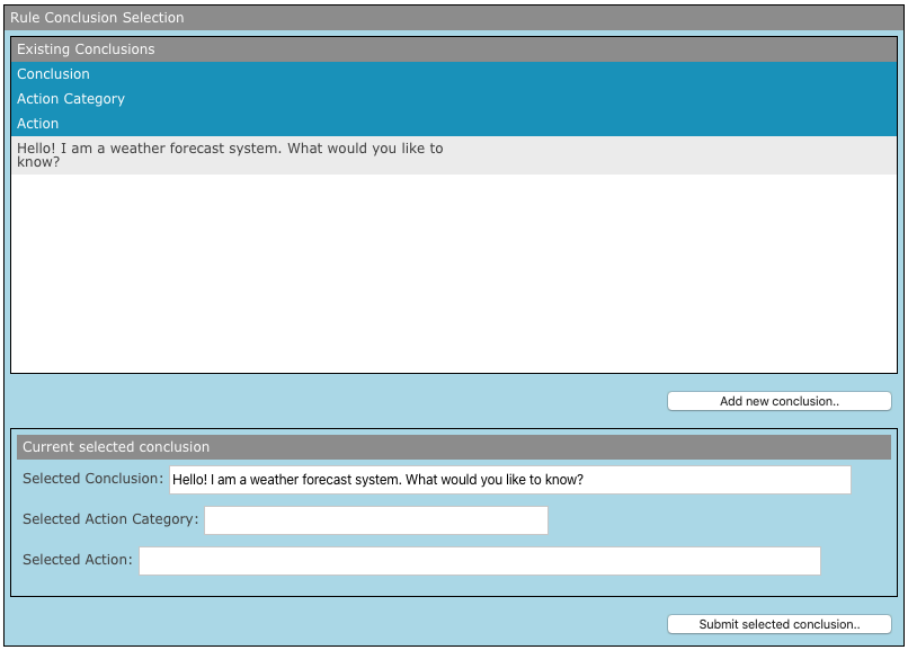
Table 3.20 – *Continued from previous page*

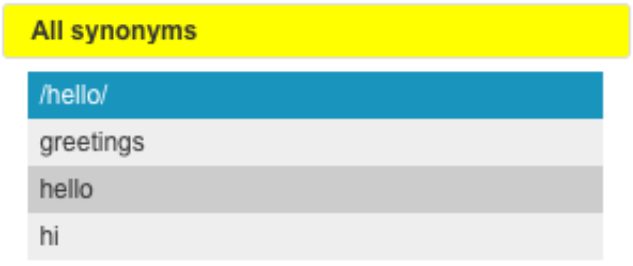
Step	Comment
3	Knowledge acquisition begins by considering the current conversational context (the last non-default satisfied rule in a previous stack frame – see Section 3.3.1), which also includes the current value of context variables. As acquisition has begun here with the default (root) context, a new rule will be added as an exception (refinement) of the default rule. 
4	The conversational author selects an existing conclusion (the rule consequent), or they create a new conclusion (as is done here). Note a simple conclusion is being added, without reference to queries or context variables. 

Continued on next page

Table 3.20 – *Continued from previous page*

Step	Comment
5	The conversational author submits the conclusion to the conclusion list and then selects it as the final rule conclusion for the rule under construction.
6	Dictionary terms and synonyms (as well as context variables) can be defined at any time prior to the conversational author defining dictionary term matching in the rule's antecedent. Note that synonyms can be maintained at any stage before or after rule construction.

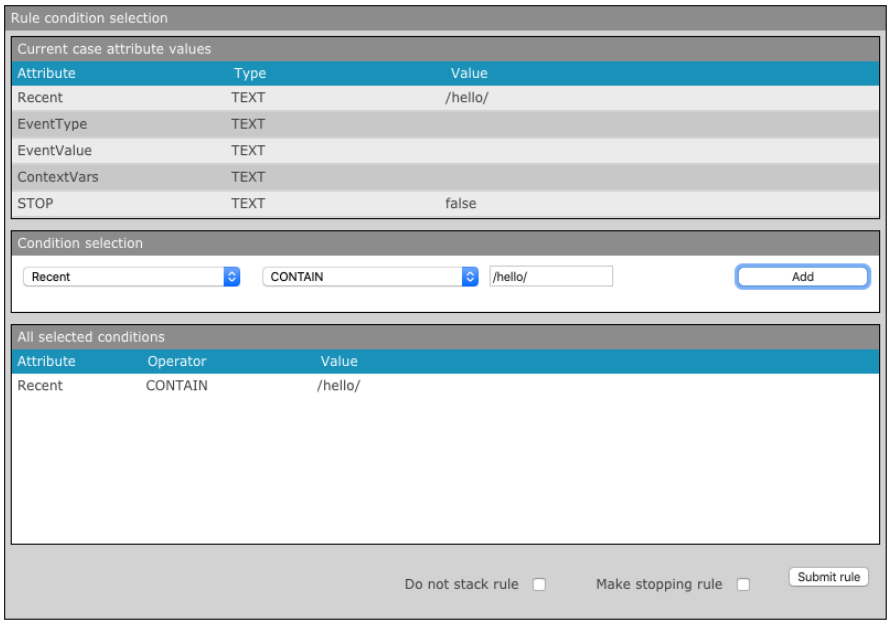


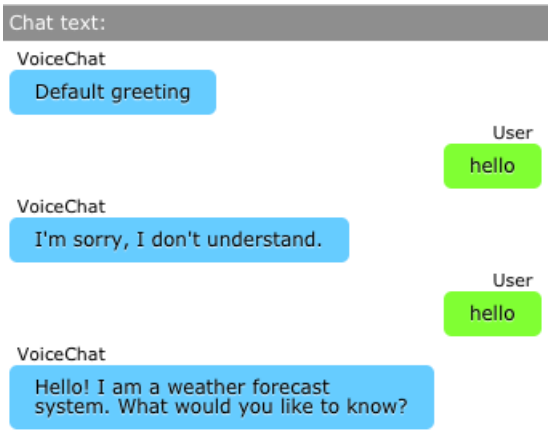


Continued on next page

Table 3.20 – *Continued from previous page*

Step	Comment
7	The new rule antecedent is defined to classify the user utterance based on the selected attributes (the example here shows the recent utterance must contain the dictionary term `/hello/`). Other attributes such as the existence or definition of context variables can be included in the antecedent, however these were not evaluated or used in the C-MCRDR CA system (such evaluation will be a component of future work).
8	The user utterance can now be repeated to show the new (correct) utterance classification – this is the system conversational reply.

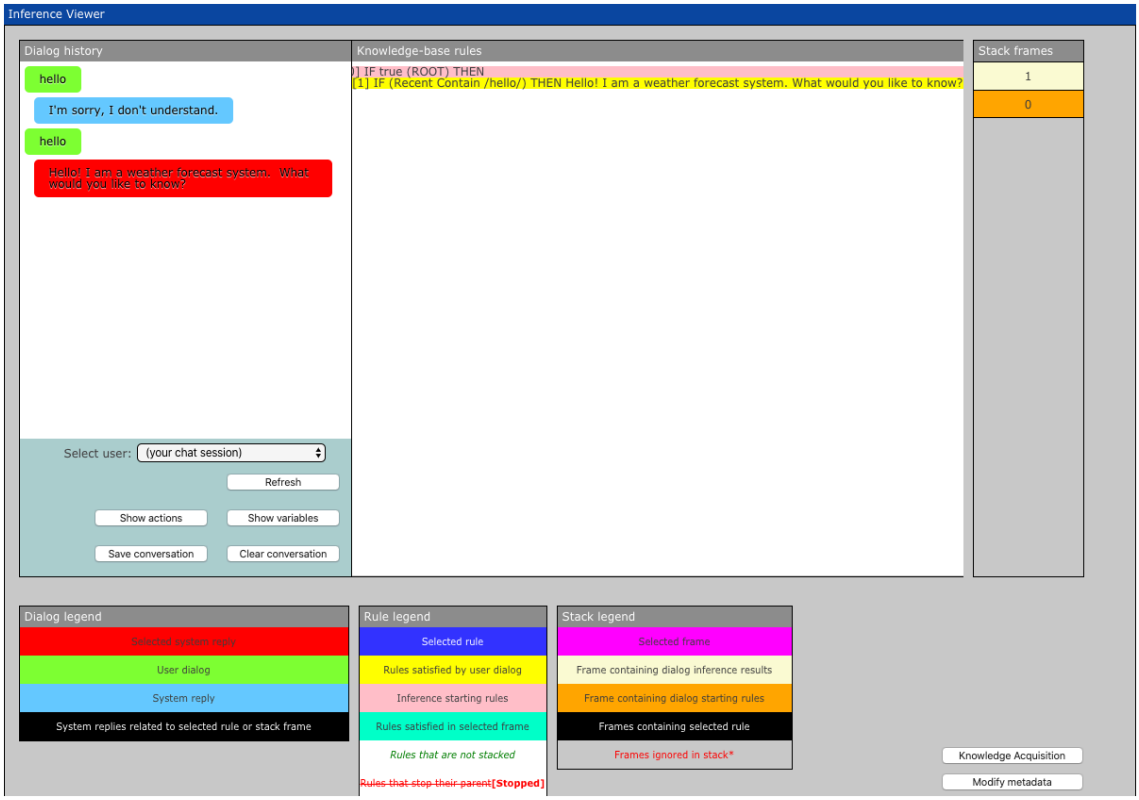




Continued on next page

Table 3.20 – *Continued from previous page*

Step	Comment
9	The new rule is now evident in the knowledge-base (the implemented administrative tool allows the conversational author to examine stack frames, conversational utterances and rules to follow the reasoning for the inference results. Selecting an utterance in the *Dialog history* will cause the associated satisfied rule(s) and stack frame to be highlighted, or vice-versa). Note that the developed tool supports multiple concurrent (but independent) inference sessions from remote web-browser clients, allowing the conversational author to select individual sessions for examination and potential knowledge acquisition.



3.3.13 C-MCRDR CA System Evaluation Methodology

A C-MCRDR CA system was implemented, with a knowledge-base that was targeted against a suitable domain. Criteria used to determine a suitable domain included requiring the existence of an *in situ* database, a domain expert with suitable knowledge of the exisiting systems, suitability of a question and answer paradigm approach, and the availability of participants for a feasibility evaluation study – a web-based documentation generation and retrieval system for course data at the author's university matched the criteria. As the author of this thesis was the original software developer for the existing documentation generation system, the author assumed the domain expert role.

The web-based documentation system is frequently accessed by undergraduate students (who are recruited as participant volunteers) and its database contains details about each *unit* taught by the ICT discipline, such as a unit's title, 6 character unit code, lecturing staff, teaching pattern, learning outcomes, assessment items and due dates etcetera. The documentation system creates a consolidated document called a *unit outline*.

This domain was then evaluated as to whether it can be complemented by a question and answer paradigm where students can ask common questions such as:

> "*Who is the lecturer for KIT101?*", and
> "*How many assignments are there for KIT102?*"

The knowledge-base was constructed with an anticipatory coverage of the key items that students refer to in the documentation system, and during evaluation, no rules were added or refined (even though C-MCRDR knowledge acquisition could have easily facilitated misclassification corrections).

3.3.13.1 User Acceptance

Participants were asked to assess both the usability of the system and the appropriateness of each of the system's responses to their questions on a 5-point Likert scale via an integrated feedback system (see Figures 3.16 and 3.17). Participant can also optionally rate the overall system effectiveness using a similar scale, and provide an optional comment (which can be a suggestion for improvement) – see Figure 3.18.

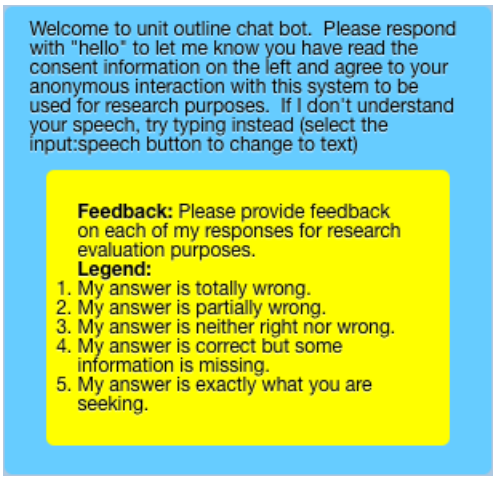


Figure 3.16: Rule feedback interface overview

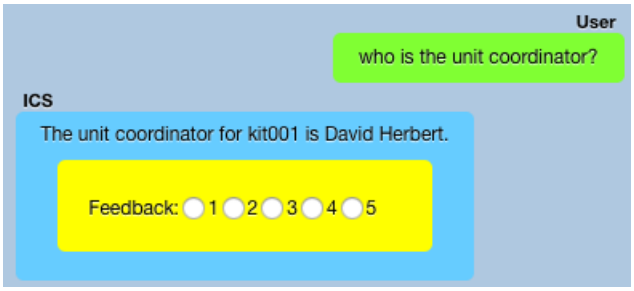


Figure 3.17: Rule feedback interface

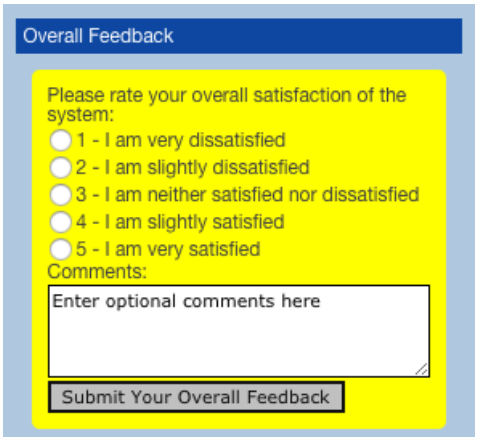


Figure 3.18: System feedback interface

3.3.13.2 Participant Recruitment

ICT students were asked to optionally volunteer to participate in the C-MCRDR CA system evaluation via:

- flyer advertisements handed out in tutorials (subject to unit coordinator consent) and posted on notice boards;
- emails sent to ICT unit coordinators requesting their help to recruit their students;
- a news item on the ICT discipline's web site requesting participation

3.3.13.3 System Evaluation Duration

Ethics approval (H0016281) was granted for the duration of the research. The major intensive period of outline reference is usually within the first few weeks of each semester.

3.3.13.4 Participation Data Logged

Logged data includes:

- the date and time;
- the participant's question;
- the system's response (including a rule trace);
- the participant's question and system feedback rating (user acceptance)

The data was stored in a file named using a non-user-identifiable 32-byte session identifier, which was generated by the participant's client web browser.

3.3.13.5 Analysis Planned

The resulting data log files were analysed using content analysis, reviewing participant ranking and comments responses, looking for usability triggers that indicate poor system performance and/or usability.

Analysis considered the following outcomes from the logged data:

- Comparison of student system ratings versus length/number of questions

- Comparison of student system ratings versus chat session duration
- Number of questions asked versus session duration
- Comparison of student system ratings based on previous system inference help response (randomised rule condition synonyms)
- Comparison of student overall system rating with variations of all the above

User acceptance was evaluated by two aspects of voluntary feedback:

(a) the usability of the entire system (a qualitative perceptive evaluation of the overall system performance); and

(b) individual system responses to inference requests.

3.3.13.6 Expected Evaluation Outcomes

Outcomes that were informed from the data analysis would be used to determine (for example):

- Knowledge-based rules that were rarely satisfied (i.e. responses never reached)
- Common synonyms used by participants that were not in the system's dictionary
- Responses that were regularly rated poorly
- Phrases used by participants that were not understood or misunderstood
- Responses that were rated highly that were very commonly encountered, perhaps helping shift focus to allow similar responses in future

3.4 Intelligent Personal Agent Methodology

This section contains the following subsections:

This methodology is aligned with SRQ4 (see Section 1.2.1):

a) *What is the best current market-leading Intelligent Personal Assistant (IPA) to leverage as a speech component of a conversational agent system in terms of ASR performance that allows the default vendor-defined conversational agent to be supplanted?*; and

b) *How can the IPA's associated ASR errors when coupled to a human-authored rule-based KBS be corrected?*

3.4.1 IPA Device Measurement

The general application development environments for current market-leading Intelligent Personal Assistant (IPA) devices such as *Google Home* and *Amazon Echo* were examined to determine how to couple them to an existing knowledge-based conversation system based on Contextual MCRDR (C-MCRDR) (Herbert and Kang, 2018).

The devices follow a workflow where they upload their compressed audio recording of a user's utterance (after a device-activating invocation word) to Google or Amazon's cloud-based clusters of servers for processing by an ASR service (Google, 2018*a*; Amazon, 2018; Hoy, 2018; de Barcelos Silva et al., 2020), (Figure 3.19, step 1). Here the utterance's speech-to-text (STT) result is then compared against a list of developer-defined sample utterances (that are mapped to corresponding *intents*) for the particular application service or *skill* in use. If a match is found, a specific intent request is then sent to the application developer's server for processing (Figure 3.19, step 2). The application developer's server then sends an appropriate conversational response back to the original ASR servers (Figure 3.19, step 5), at which point a final text-to-speech (TTS) response is directed to the assistive device (Figure 3.19, step 6).

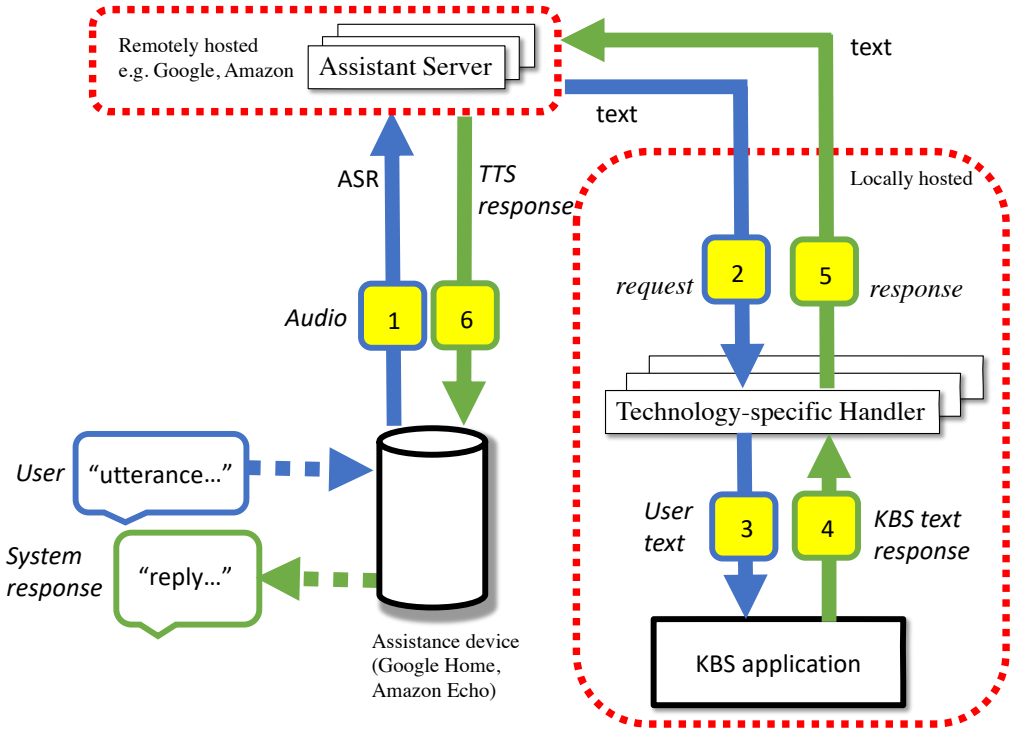


Figure 3.19: Assistive device system architecture

The device's standard workflows are not tightly coupled with the use of separate conversational agents that facilitate incremental knowledge acquisition (conversational knowledge). The vendor workflows require a developer to predefine all conversational intents and sample utterances – these samples cannot be modified dynamically. Instead, in this environment, new conversational knowledge has to be added offline (by a developer) and the conversation application has to be re-built and re-published. This is problematic and inefficient if IPA devices are integrated with the C-MCRDR CA system from the first phase of the thesis' methodology; the overall RDR methodology allows the domain expert (conversational author) to add new rules to the knowledge-base at any time, which is not supported by the IPA device workflow. This dynamic nature of new utterance coverage, coupled with the possibility that utterances should only be recognised in certain (rule-based) contexts, means offline (and duplicated) utterance predefinition is not suitable (nor does it leverage the advantages of MCRDR-based knowledge acquisition).

The solution is to simplify the Google- or Amazon-hosted skill to pass the *original* STT utterance directly to a downstream client service for processing, bypassing matches against the statically defined sample intents. The user's raw speech-to-text utterance is then passed to the end service KBS application (Figure 3.19, step 3) to determine an appropriate conversa-

tional response (Figure 3.19, step 4). This process however exposes the *raw* server-based ASR recognition performance – recognition errors would normally be obscured due to most-probable intent matching, and as a consequence, the raw error-prone ASR results are received at the conversation system interface (Figure 3.19, step 2).

The ASR recognition rates are compared for two contemporary, market-leading commodity devices – the Google Home and the Amazon Echo; both device's developer environments allow for the capture and subsequent processing of the intermediate raw STT results by the downstream server. A recently released (February, 2018) equivalent product, Apple's *HomePod* currently does not expose the same level of access to the raw STT results in their SiriKit API (Apple, 2018) and consequently this device was not tested.

Once the best performing device is determined, a method for correcting common mis-recognised terms *after ASR* is needed. The approach taken is to treat the IPA as a black box, correcting terms using a rule-based paradigm; mis-recognised terms are pattern-matched against domain expert-defined rules before the final, pre-processed corrected term is used for inference with the conversation system. As mentioned in Section 2.3.3.1, a future consideration is a generalisation of the rule-based correction to include potential candidate word replacements based on threshold string edit distance (Levenshtein, 1966).

3.4.2 Recognition Performance Improvement

After a device is selected based on its isolated word and sentence recognition performance, the post-ASR recognition rate are then improved so that the most accurate utterance is presented to the underlying KBS prior to inference. A raw utterance is defined to be subject to rule-based corrections that are determined by a replacement policy. The policies are defined by the *location* in the KBS C-MCRDR decision tree that specifies when corrected replacements occurs – this is directly related to the current conversational context (topic) being discussed; two policies are defined – *Global ASR correction* and *Regional ASR correction*. These are forms of manual, supervised correction editing (Ringger and Allen, 1996; Mangu and Padmanabhan, 2001), however, a *Global ASR correction* policy could be implemented algorithmically for automatic or semi-automatic rule insertion if statistical means were considered (Bassil and Semaan, 2012; Sarma and Palmer, 2004), or approximate string matching (see Section 2.3.3) techniques were used. For word error rate (WER) improvement, the evaluation data (a constrained vocabulary of isolated words) was presented to the system in an explicit known sequence, and the generated transcriptions can be matched on a one-to-one basis. This meant global ASR correction replacement rules could have been created algorithmically as each transcription result is captured by

the software. This was not implemented in the evaluation system however, instead, correction rules were manually inserted between each round of correction – please see Section 3.4.3 for the evaluation environment details.

Global and regional correction policies both define correction rules between each iteration round of correction, but due to unavoidable variability in pronunciation, in each round only a subset of defined correction rules are actually applied. The *effort* of the correction method is defined to be the number of corrections applied divided by the number of corrections defined. Ideally the ratio should be 1 (only a minimum number of corrections need to be defined that are all then applied) and a ratio of zero indicates maximum effort (a number of corrections are defined but none are applied). This is evaluated in Sections 4.3.6.1.3 and 4.3.6.2.3.

3.4.2.1 Global ASR Correction

Global ASR correction is the simplest approach and it is currently the policy implemented in the C-MCRDR CA system development code that initially corrects browser-based speech input. Original lexical or phrasal utterances are pattern-matched against a set of correction rules (RE_G) that are defined by the domain expert as a pre-processing step for speech input, and they are correctively applied to STT inputs in order. Corrections are ignored when the user interface to the conversation system uses text. The global ASR correction definitions can be found in the C-MCRDR CA system methodology, Section 3.3.9.1.

A global ASR correction policy is only suitable when the raw utterance term to be replaced does not occur in any rule antecedent (i.e. terms are not replaced that may already satisfy any rules in the knowledge-base decision tree).

3.4.2.2 Regional ASR Correction

The regional ASR correction proposal is more interesting as it is defined to apply to a constrained subset of rules in the decision tree where corrections may occur. The region is specified by a starting rule number, r_r and it is extended to the subtree containing all children from rule r_r onwards. Regional ASR corrections allow for more contextual utterance corrections, including corrections that would otherwise potentially cause erroneous satisfaction/dissatisfaction of rules not included in the region. In other words, corrections are only processed when the candidate components of an utterance are determined to not occur within the region (this can be overruled by the definition of corrections for a sub-region). The regional correction policy is facilitated by the altered, stack-based inference process in the C-MCRDR CA system. By starting inference at specific, satisfied rules it is possible for an utterance to potentially be presented to any sub-

tree or region in the knowledge-base, bypassing all parent rules. It is at this point regional ASR corrections can be processed (and rolled-back if no rules are satisfied).

In contrast to the set of global ASR corrections (RE_G), RE_r is defined as an ordered set of user-defined regular expressions e''_i with corresponding actions (text replacements) that are applied to the ordered set of rule nodes T_r (a *region*) from rule r_r to all children:

$$RE_r = \{e''_i \mid 1 \leq i \leq m\} \tag{3.43}$$

where m is the total number of region-specific corrections. t''_0 is specified as the raw utterance and t''_m as the final (post-replacement) utterance after all regional corrections have occurred, with t''_m defined as the recursive process:

$$\begin{aligned} t'_1 &= e''_1(t''_0) \\ t'_m &= e''_m(t''_{n-1}) \end{aligned} \tag{3.44}$$

where $e''_i(t)$ is the application of the text correction for regular expression e''_i to text t. Note if $r_r = r_0$ then this is a global ASR correction policy.

Assuming a sub-region T_x is defined with respect to parent region T_a by:

$$T_x \subseteq T_a \mid x \geq a \tag{3.45}$$

then corrections specified for the sub-region take priority over those specified for the parent region (rule-specific corrections always take precedence over parent regional corrections). *Implementation of a regional policy replacement scheme will be the subject of future work, however introspective evaluation of a regional replacement policy can be found in Section 4.3.6.2.*

3.4.3 Software Environment and Knowledge-base Configuration

The raw STT result is captured in the developed skill (in the vendor's authoring environment) by using a single intent with a single LITERAL slot type (Alexa Skills Kit, (Amazon, 2018)), and a single default intent (Google Actions SDK, (Google, 2018*a*)). The text result is then forwarded to the downstream application server which runs two custom application handler services (for Alexa and Google Assistant requests). The appropriate application then establishes an HTTP session to send the utterance to the C-MCRDR CA system application, and it receives the conversational result by inference as a reply. For testing the STT results, the C-MCRDR CA system is configured with a very simple one-rule knowledge-base (see Table 3.21).

A lexical or phrasal paraphrasing dictionary term `/word/` is defined by a regular expression to match any input terms. Coincident with this, a contextual variable `@word` is defined – it

Table 3.21: KB definitions for ASR correction

Type	**Term**	**Definition**
dictionary	`/word/`	`/re: ^.*$`
variable	`@word`	`/word/`
rule	R_1	*Condition*: `input contains /word/` *Conclusion*: <LIT>@word</LIT>

will be assigned the text that specifically matches the dictionary term. Finally, the single (non default) C-MCRDR rule is satisfied if the dialog input attribute contains any lexical or phrasal terms that match the dictionary term `/word/` (any line will match). The rule's conclusion is to simply return the literal value of the `@word` variable - in other words, an exact copy of the input. This means, for the evaluation, the KBS returns a copy of its input (which is then eventuality vocalised by the assistive device). The key point here is what is received is logged by the KBS and then compared to the original vocalised source utterance. This can then validate if the assistive device's virtual assistant correctly recognised the original vocalisation. The *Technology-specific Handlers* developed for each device in Figure 3.19 could have directly logged and returned the received utterance as a response to the skill request (without KBS interaction), however, the utterance is passed to the KBS as it is here recognition terms are corrected prior to requesting inference.

3.4.4 IPA Word Data Source

The Amazon Echo and Google Home were evaluated by their respective agents, Alexa and Google Assistant (which are then constrained in this context by the IPA hardware configuration, such as number and type of far-field microphones, acoustic conditions and so on) with two data sources:

1. A lemmatised frequency list (Kilgarriff, 2006) for 6318 words extracted from the British National Corpus (BNC) (Natcorp, 2018) that have a frequency of 800 or more. Evaluation data was then sorted into twelve categories based on word letter count (ranging from two to thirteen letters) from the list. Within each category, the first 100 words were then chosen (although the 2-letter and 13-letter categories were limited to 25 and 71 words respectively). In total 1096 words were tested for ASR recognition. Each word list was also then annotated with their associated *rank*, for example, the rank 1 word is '`the`' (the highest frequency word chosen), and '`supplementary`' was the lowest frequency word (which is ranked 6261);

2. Seven 100-sentence lists ranging from 4 words to 10 words per sentence. Sentences were generated from a Markov chain trained by the "*A Christmas Carol*" (Dickens, 1843) text using an established Markov chain generator application (Singer-Vine, 2014).

The assumption was made that there is a reasonable probability the Google and Amazon ASR acoustic and linguistic models are trained using high-frequency English words which informed the choice of data described in item (1) on page 131. As opposed to just simply testing the top number (for example, the first 1000) of highly-frequency words, categorising the test data by letter count will establish if there is any relationship between word length and recognition rate under the hypothesis that longer length words are more likely to contain more syllables (and thus phonemes) to recognise. All words in the evaluated dataset were manually annotated with their associated syllable count for comparison purposes.

The sentence data described by item (2) above was used to determine if there was any relationship between sentence length and recognition rate – a reasonable expectation is the addition of more context (words) will improve the recognition rate.

Each IPA device were evaluated under the same environmental conditions (room, acoustics, speakers and proximity to the device's microphones) between tests. Word lists and sentences were spoken both by the thesis author (an Australian English native speaker) and the Australian *Karen* voice included with Apple macOS X High Sierra 10.13 operating system. The spoken word rate was as invariant as possible in order to remove effects of variations in ASR error (Jeanrenaud et al., 1995).

Word Error Rate (WER) (Park et al., 2008; He et al., 2011; Bisani and Ney, 2004; Raghavan and Allan, 2005; Fiscus et al., 2006), detailed in Section 2.3.4.1, is used as the main metric in determining ASR performance between IPA devices and different source speakers (human or *Karen*, a computer-generated voice included with Apple macOS X High Sierra 10.13 operating system). In the isolated word category evaluation, each sentence consists of one word, and as WER is measured using *edit distance* (Levenshtein, 1966) *at the word level* (i.e not at the individual character level), calculations are performed by determining if the ASR response is a direct match, or not. In this case, a word substitution, deletion, or insertion have all the same effect for single-word sentences - the response either matches the source word or it does not.

Unrecognised vocalisations (i.e. the IPA device failed to recognise a word) were repeated to a maximum of three times (this was more frequent for shorter words with the computer-simulated *Karen* voice). Initially, recognition was deemed successful when the recognised response was also a homophone of the source word. This is because in the testing environment it is reasonable to allow homophones as a word cannot be distinguished from the original source intent

without any surrounding context. In a real environment differentiating between actual intent and homophones may be critical. Homophones are also corrected by the correction policy steps where necessary, and in evaluation for WER improvement by global and regional corrections, homophones of source words are thus treated as distinct errors in recognition. For word rank and isolated word tests, homophones are treated as the same as the source word, so the WER values for these tests are slightly smaller than the sentence or WER improvement results – please refer to Section 4.3.3.1 more more detail.

3.4.4.1 Data Preprocessing

Prior to testing IPA performance for generated sentences (see item 2 above), each sentence was examined and manually preprocessed to replace text or anachronistic words or proper nouns that are not in *common usage*, for example, "*Mrs. Fezziwig*" or "*Mrs. Cratchit*" were replaced.

Chapter 4

Evaluation and Discussion

> "*For seven and a half million years, Deep Thought computed and calculated, and in the end announced that the answer was in fact Forty-two - and so another, even bigger, computer had to be built to find out what the actual question was.*"
> – Douglas Adams (Adams, 1980)

This chapter contains the following sections:

4.1 Introduction

The results in this section detail the effect of defining and applying a C-MCRDR conversation system in two aspects:

1. C-MCRDR CA System Evaluation (please see Section 4.2) – a C-MCRDR CA system is evaluated (defined in Section 3.3 and specifically in Section 3.3.11) against a pedagogical domain; the domain's effective rule-count reduction compared to an MCRDR approach is examined, together with the rules satisfied, overall system performance (in terms of appropriate responses) and user acceptance following the evaluation trial.
2. Intelligent Personal Assistant Evaluation (please see Section 4.3) – the Automatic Speech Recognition (ASR) performance is evaluated for two current market-leading commodity Intelligent Personal Assistants (IPA) smart speakers, the *Google Home* and the *Amazon*

Echo in order to determine the best device to use as a speech-enabling interface to the C-MCRDR CA system. Evaluation consisted of comparing the final speech-to-text results with the original source words and sentences, and the effects of different word and sentence attributes on speech recognition performance are determined. Two ASR error correction schemes, *global* and *regional* correction were also evaluated – please see Section 3.4 for the methodology.

4.2 C-MCRDR CA System Evaluation

Section 3.3 in Chapter 3 details the methodology associated with the definition, implementation and evaluation configuration of the C-MCRDR CA system. The results and analysis of the evaluation are detailed in this section and as previously noted they answer sub-research questions (see Section 1.2.1):

- SRQ1 *How can conversational context be maintained in a human-authored rule-based conversational agent system?*
- SRQ2 *What is a pattern-matching approach that does not require complex scripting and programming skills or knowledge of formal grammatical syntax that can provide high levels of system performance?*
- SRQ3 *What augmentations to the MCRDR rule-based KBS methodology will produce conversational agent systems that can achieve high levels of system performance?*

This section consists of the following sub-sections:

4.2.1 Evaluation System Knowledge-base Data Summary

The C-MCRDR CA system's knowledge-base data can be found in Appendix A.1 and A.2. To summarise:

- Appendix A.1 Speech Corrections:
  - Speech-to-text corrections – Table A.1;
  - Text-to-speech corrections – Table A.2;
- Appendix A.2 C-MCRDR CA System Knowledge-base:
  - Knowledge-base rules – Table A.3;
  - Lexical Paraphrase terms – Table A.4;
  - Database queries – Table A.9;
  - Context variables – Table A.12

System evaluation feedback data can be found in Appendix A.3:

- Rule feedback scores – Table A.13;
- Rule feedback sessional scores – Table A.14;
- Cat1 session feedback scores – Table A.15;
- Cat2 session feedback scores – Table A.16;
- Cat3 session feedback scores – Table A.17;
- Cat4 session feedback scores – Table A.18;
- Sessional inference requests by category – Table A.19

4.2.2 Evaluation Trial Participants

Participants were recruited from undergraduate students studying ICT units to use the prototype conversation system associated with the unit outline domain, but no specific unit or student cohort was targeted in order to reduce selection bias.

Table 4.1: Global derived data from participation trial

Ref	Item	Value
1	Total number of sessions	41
2	Longest session	1011s
3	Shortest session	32s
4	Mean session length	278s
5	% of sessions with per-question feedback	58.5%
6	% of sessions with overall system feedback	51.2%
7	Total number of inference requests (questions)	478
8	Number of questions with client feedback	127
9	% of questions with user feedback	26.6%
Per-session data		
10	Mean number of questions	11.7
11	Mean number of correct questions	6.1 (52.7%)
12	Mean number of misinterpreted questions	0.5 (4.4%)
13	Mean number of valid questions with default response	1.8 (15.3%)
14	Mean number of invalid questions with default response	3.2 (27.6%)
15	Mean of sessions that selected document link	7 (17.1%)

Table 4.1 provides a summary of the participation trial sessional data. Item 4 is large due to one outlier session (due to an idle time between questions of 7.6 minutes). As participation, including feedback, was voluntary, only slightly more than 1 in 4 question responses had feedback given (Item 9). The system interface also provided a URL link each time a different unit code was requested – this link referred to the original unit outline PDF document (Item 15) that is generated by the unit outline system. Interestingly, this was rarely used.

4.2.3 Rule-count Reduction

Results from this section help to answer research question SRQ3 (see Section 1.2.1).

The ICT Discipline at the author's university delivered 34 distinct units that were detailed in the discipline's unit outline documentation system. Each unit required generation of a unit outline document during the data-gathering phase of this research. In total there were 140 individual assessment tasks when combining all of the unit information (giving an average of 4

assessment items per unit), 20 coordinating lecturers and 23 distinct teaching teams (consisting of 1 or 2 people). In terms of database querying, every unit outline document refers to 28 distinct database fields, and every assessment item refers to two additional database fields.

Table 4.2 depicts the affinity factor summarisation as defined in Section 3.3.7, Equation 3.23. Please recall an affinity set is defined as a set of rules that can be replaced by a single C-MCRDR rule. *Rules* in this table specify the equivalent *name* of an MCRDR rule. For example, the evaluation pedagogical domain consisted of 34 distinct unit codes, so MCRDR rules that capture user *unitcode* intent consist of 34 rules, $unitcode_1$ to $unitcode_{34}$. The final evaluation knowledge-base contains 36 C-MCRDR rules in total (see below – a few rules are not mentioned in Table 4.2, but they include `METAHELP`, `METAWHERE` and an "out-of-scope" unit code checking rule). The Grand Total for MCRDR in Table 4.2 in effect shows the minimum number of rules (prior to the adoption of queries and context) needed to provide the classification outcomes using the standard MCRDR approach. Without post-inference query referencing, the contents of the referential database would subsequently need to be flattened and then directly encoded by one-to-one MCRDR rules in the knowledge-base.

The Grand Total (N') for the number of C-MCRDR rules in Table 4.2 is derived from Equation 4.2.3 in Section 3.3.7:

$$N' = \sum_{i=1}^{L} N'_i$$

Given $L = 5$, each N'_i is the size of each affinity set (Equation 3.26) and it then follows that:

$$\begin{aligned} N' &= |A_1| + |A_2| + |A_3| + |A_4| + |A_5| \\ &= 1 + 1 + 1 + 21 + 9 \\ &= 33 \end{aligned}$$

So through the evaluation of the affinity sets, it was determined that 33 C-MCRDR rules are the equivalent to 1158 MCRDR rules in this domain.

The *unit outline* domain is relatively simple, so such a high MCRDR rule count as shown in Table 4.2 is excessive and in general, a standard MCRDR approach would not be used without modification or optimisation. The C-MCRDR definition of 36 rules (33 + 3), compared to the equivalent MCRDR knowledge-base rule count of 1161 (1158 + 3), is a 96.9% rule count reduction. For illustrative purposes, part of the knowledge-base from both approaches is partially visualised in Figures 4.1 and 4.2, with only 4 (of 36) attributes of the dialog shown (the full list of rules defined for the C-MCRDR CA system can be found in Appendix A.2, Table A.3).

In this illustrative example it is observed that there is a signification rule count reduction (C-MCRDR – 5 rules, MCRDR – 137 rules). Example conversational samples comparing the two approaches can be seen in Table 4.3.

Table 4.2: MCRDR Affinity Factors

AF	**AS**	**Rules**	**MCRDR**	**C-MCRDR**
A_1	$\alpha_{1,1}$	$\{root\}$	1	1
A_2	$\alpha_{2,1}$	$\{hello\}$	1	1
A_3	$\alpha_{3,1}$	$\{unitcode_1,\ldots,unitcode_{34}\}$	34	1
A_4	$\alpha_{4,1}$	$\{coordinator_1,\ldots,coordinator_{34}\}$	34	1
	$\alpha_{4,2}$	$\{lecturer_1,\ldots,lecturer_{34}\}$	34	1
	$\alpha_{4,3}$	$\{assessment_1,\ldots,assessment_{34}\}$	34	1
	$\alpha_{4,4}$	$\{unitdetail_{1,1},\ldots,unitdetail_{1,34}\}$	34	1
	$\alpha_{4,5}$	$\{unitdetail_{2,1},\ldots,unitdetail_{2,34}\}$	34	1
	...	...	...	...
	$\alpha_{4,21}$	$\{unitdetail_{18,1},\ldots,unitdetail_{18,34}\}$	34	1
		Total	**714**	**21**
A_5	$\alpha_{5,1}$	$\{coordinator_1detail_1,\ldots,coordinator_{34}detail_1\}$	34	1
	$\alpha_{5,2}$	$\{coordinator_1detail_2,\ldots,coordinator_{34}detail_2\}$	34	1
	$\alpha_{5,3}$	$\{coordinator_1detail_3,\ldots,coordinator_{34}detail_3\}$	34	1
	$\alpha_{5,4}$	$\{coordinator_1detail_4,\ldots,coordinator_{34}detail_4\}$	34	1
	$\alpha_{5,5}$	$\{coordinator_1detail_5,\ldots,coordinator_{34}detail_5\}$	34	1
	$\alpha_{5,6}$	$\{lecturer_1detail_1,\ldots,lecturer_{34}detail_1\}$	34	1
	$\alpha_{5,7}$	$\{lecturer_1detail_2,\ldots,lecturer_{34}detail_2\}$	34	1
	$\alpha_{5,8}$	$\{lecturer_1detail_3,\ldots,lecturer_{34}detail_3\}$	34	1
	$\alpha_{5,9}$	$\{assessment_1item_1,\ldots,assessment_{34}item_1,$	34	
		$assessment_1item_2,\ldots,assessment_{34}item_2,$	34	
		$assessment_1item_3,\ldots,assessment_{34}item_3,$	34	
		$assessment_1item_4,\ldots,assessment_{34}item_4\}$	34	1
		Total	**408**	**9**
		Grand Total	**1158**	33

AF = Affinity Factor, AS = Affinity Sets, MCRDR = rule count, C-MCRDR = rule count

Table 4.3: Comparison user dialog (1: C-MCRDR, 2: MCRDR)

Question	R#	Response
1: *I'd like to talk about KIT101*	$\{R_1\}$	We're discussing KIT101
2: *I'd like to talk about KIT101*	$\{R_1\}$	Programming
1: *Who is the coordinator?*	$\{R_2\}$	The coordinator is
2: *Who is the KIT101 coordinator?*	$\{R_1\}$	Dr Jane Smith
1: *What are their contact details?*	$\{R_3\}$	Jane Smith: extension
2: *What are the KIT101 coordinator contact details?*	$\{R_3\}$	x9992, jsmith@x.y.edu
1: *What's the assessment pattern?*	$\{R_4\}$	The unit has 7 assignments
2: *What is the KIT101 assessment pattern?*	$\{R_4\}$	
1: *Let's discuss KIT371*	$\{R_1\}$	We're discussing KIT371
2: *Let's discuss KIT371*	$\{R_{133}\}$	Algorithms
1: *Contact details?*	$\{R_3\}$	John Jones: extension
2: *What's the KIT371 coordinator contact details?*	$\{R_{135}\}$	x9971, jjones@x.y.edu

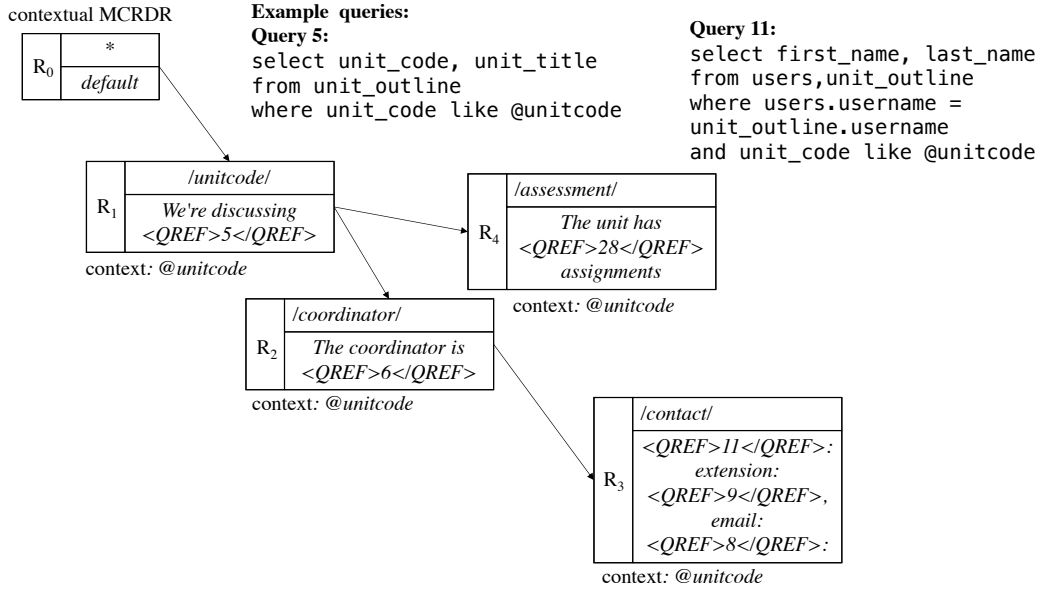


Figure 4.1: Example C-MCRDR KB for conversation system

4.2.4 Rules Satisfied During Evaluation

Results from this section help to answer research questions SRQ1 and SRQ3 (see Section 1.2.1).

The participation trial inference results (rules satisfied, number of inference requests per session etcetera) were logged and analysed.

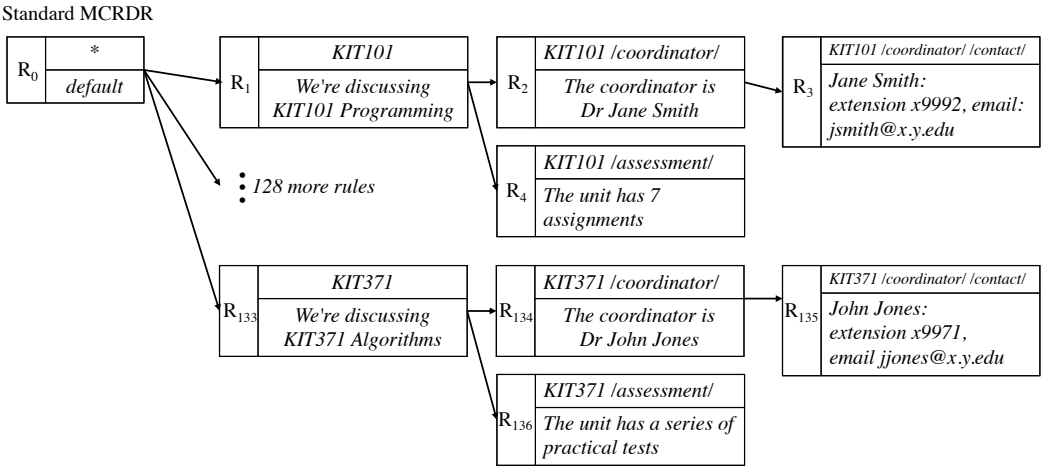


Figure 4.2: Example MCRDR KB for conversation system

Figures 4.3 and 4.4 shows the sessional frequency of satisfied rules from the trial, and Figures 4.5 and 4.6 given an indication of the average rule depth achieved across all sessions and the average depth limited to appropriate system responses respectively (cat1 and cat2 categories are defined in Table 4.6 in Section 4.2.7. Both categories refer to valid user inference requests, but the former refers to valid system responses and the latter refers to misinterpreted system responses). Figure 4.7 indicates the actual depths of all rules in the established knowledge-base and simple inspection indicates the knowledge-base has an average rule depth of 4. In this domain, rules at depth 4 are associated with simple factual data for single units. Rules at depth 5 deal with multiple data items associated with individual assessment tasks, lecturer details etcetera. Although the knowledge-base depth is relatively shallow, and as each satisfied rule in evaluation generally matches a single term in a single source utterance, the combination of Figures 4.3, 4.4, 4.5, 4.6 and 4.7 show participant sessions do reach deeper rules. This can only occur if their context has been sustained between utterances (via the stack based approach detailed in Section 3.3.1). These results then help answer Research Question SRQ1 *How can conversational context be maintained using a constrained NL interface?* (see Section 1.2.1).

4.2.4.1 Common Rules Satisfied

The ten most-frequently satisfied rules are ranked and shown in Table 4.4, highlighting the most frequent pedagogical data requested by the participants is *how a unit is taught* (rule 17), *who are the lecturers?* (rule 8) and *how many assessment items are there?* (rule 25). Examination of the most frequent rules satisfied, together with the potential user queries that could not be answered satisfactorily in this manner can greatly assist the domain expert with future

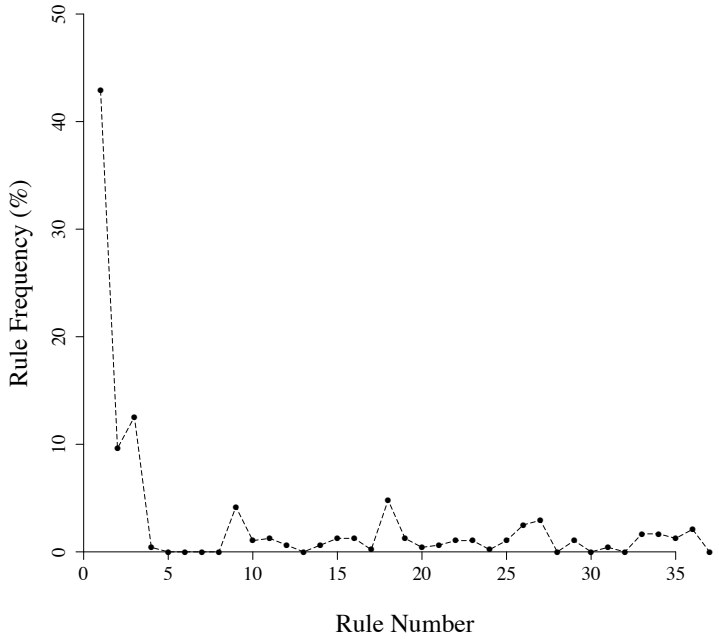


Figure 4.3: Inferred rule frequency

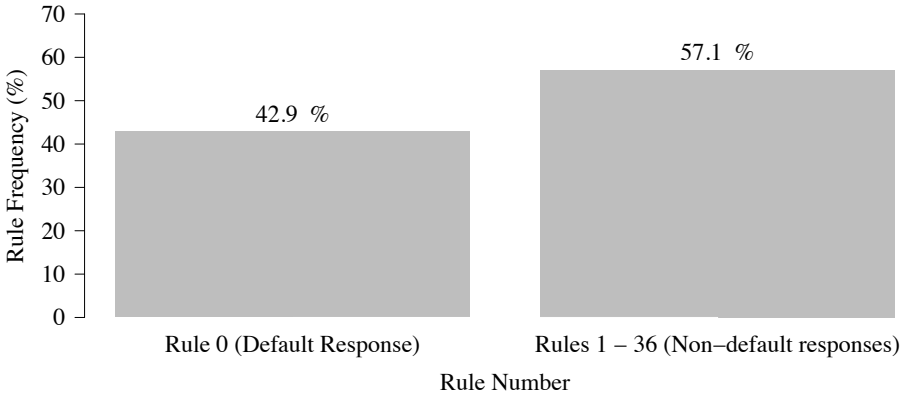


Figure 4.4: Inferred rule frequency (clustered)

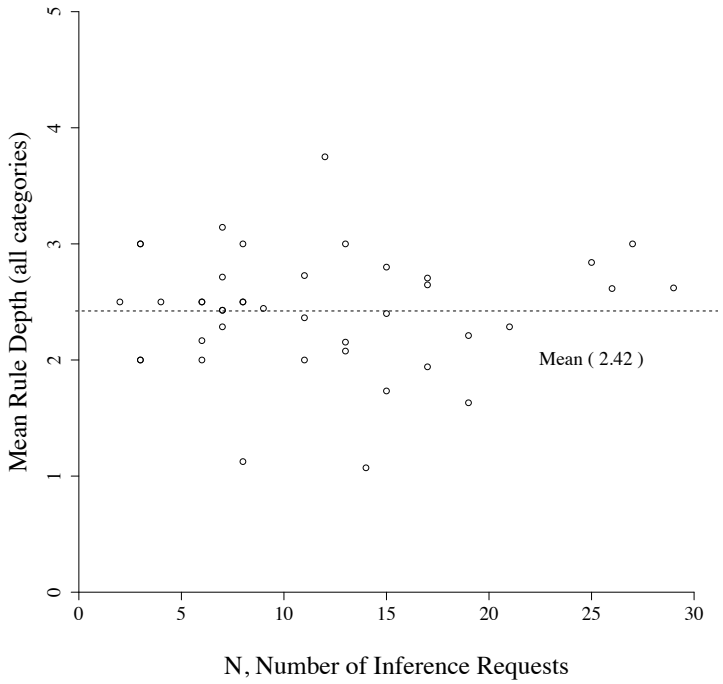


Figure 4.5: Average rule depth $\mu = 2.42, \sigma = 0.51$

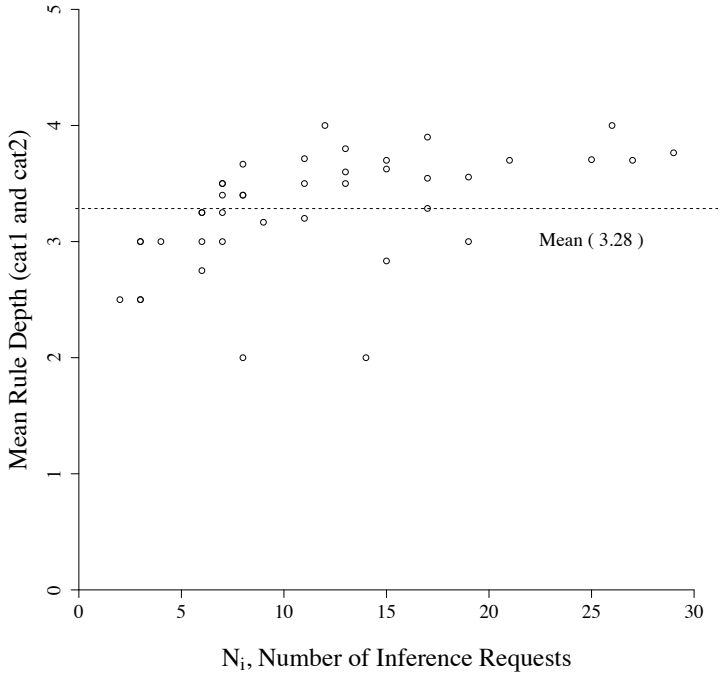


Figure 4.6: Average rule depth (appropriate responses only) $\mu = 3.28, \sigma = 0.49$

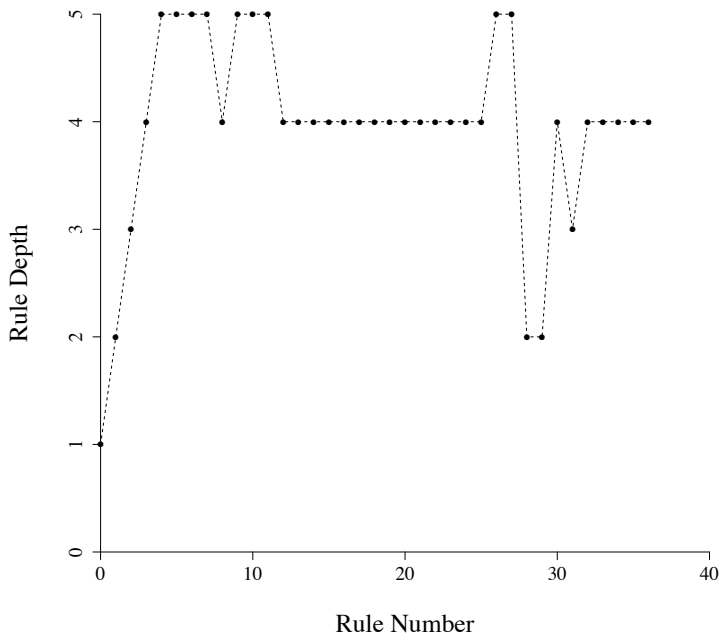


Figure 4.7: Inferred rule depth

Table 4.4: Top 10 satisfied rules

#	**Freq**	**Rule**	**Description**
1	38.93%	0	Default (root) rule
2	15.80%	2	Specify a unit code
3	13.97%	1	hello – indicates readiness
4	4.57%	17	Teaching mode (face-to-face lectures,online etc)?
5	3.91%	8	Teaching staff contact details?
6	2.42%	25	How many assessment items?
7	2.00%	26	Assessment item details?
8	1.81%	15	Attendance requirements?
9	1.58%	32	Is there an exam?
10	1.55%	35	When is a specific item due?

rule development (with incremental rule additions made possible via the C-MCRDR knowledge acquisition methodology). Greater conversational topical context (and satisfied rule depth) may possibly be achieved, together with higher levels of user satisfaction, although it must be noted no additional rule maintenance was performed during the evaluation trial.

4.2.4.2 Rule Satisfaction Zipf Relationship

If the satisfied rules from Figure 4.3 are ranked and ordered by decreasing frequency, R_f, and plotted by log of frequency by log of rank (R_r), it yields Figure 4.8, where the resulting best fit linear model ($R^2 = 0.94$, $p < 2 \times 10^{-16}$) is indicative of the power relationship:

$$R_f = 186.26 \times \frac{1}{R_r^{\alpha}} \qquad \alpha = 1.37 \tag{4.1}$$

Plotting the ordered frequency R_f directly against rank R_r in Figure 4.9, shows the fit of the power relationship from Equation 4.1 to the actual data.

This relationship is of interest as it is close to Zipf's law (Zipf, 1949), where it was observed (and subsequently shown for many other non-linguistic phenomena) that the frequency of words (L_f) in written texts is inversely proportional to their ranking (L_r):

$$L_f \propto \frac{1}{L_r^{\alpha}} \qquad \alpha = 1$$

This result is perhaps intuitive as the rule's antecedents are closely aligned with the pattern matching of words, albeit in a highly reduced and constrained domain-specific dictionary subset of English.

4.2.5 Effective Query Execution

Results from this section help to answer research question SRQ3 (see Section 1.2.1).

The potential rule-count reduction exhibited during actual evaluation of this domain (in comparison to standard MCRDR) can be seen in Figure 4.10, where the number of distinct queries ($\text{SQL}_{distinct}$) that are referred to by C-MCRDR is plotted against the number of inference requests (N_i) (the data is averaged for each distinct N_i). Overlaid is the equivalent count of standard MCRDR rules ($MCRDR_s$) that would be required to achieve the same inference results. Overall, to facilitate equivalent complete domain coverage, the knowledge-base would have to define 1161 standard MCRDR rules (as previously shown by considering affinity sets in Table 4.2). However, during evaluation each participant session only accesses a reduced subset of the domain knowledge-base ruleset (as they do not ask all possible questions). For instance,

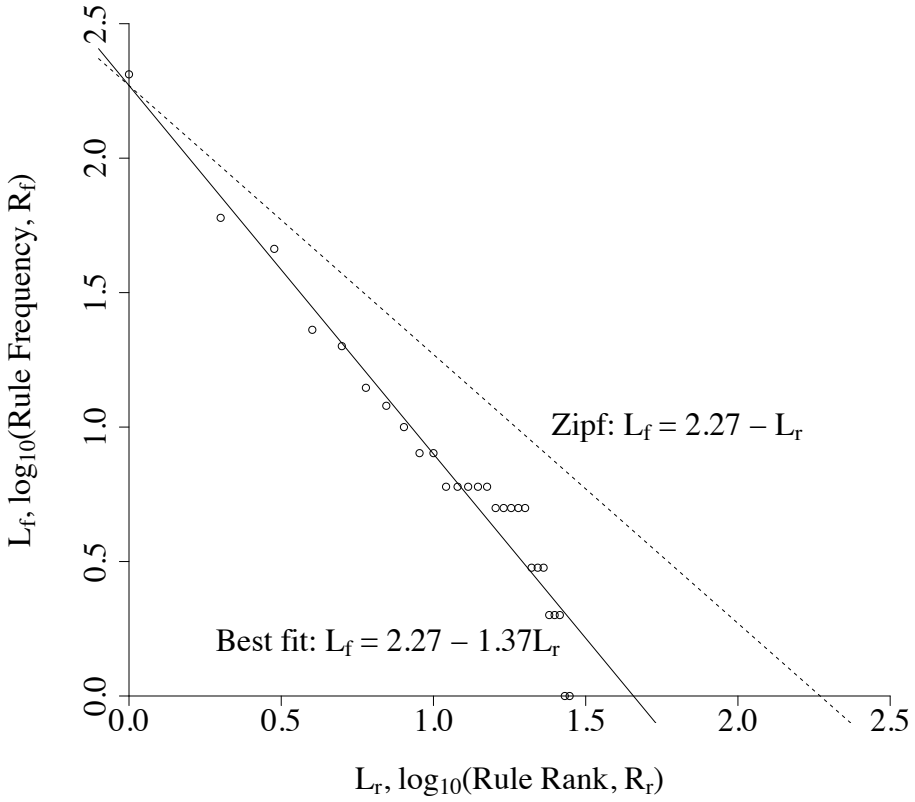


Figure 4.8: Log rule frequency distribution

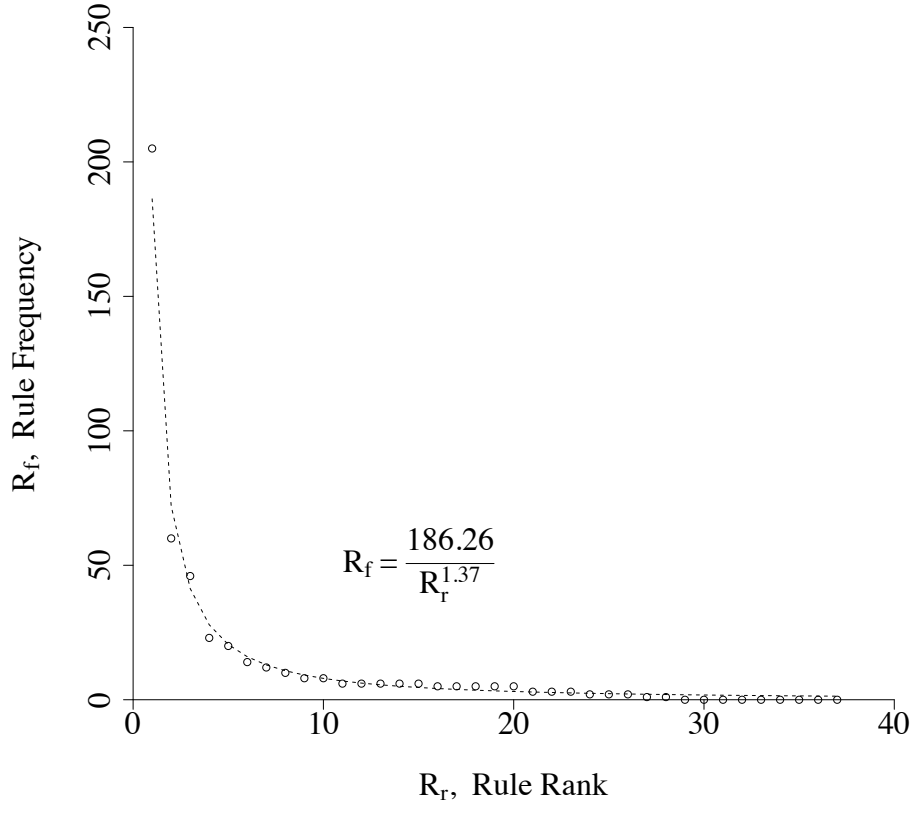


Figure 4.9: Rule frequency by rule rank

the highest number of *unitcodes* referred to in an individual session was 6 (at N_i=25,26). Please note the significant outlier at $N_i = 14$ – this anomalous session is discussed in later sections, for example, Section 4.2.8.1.

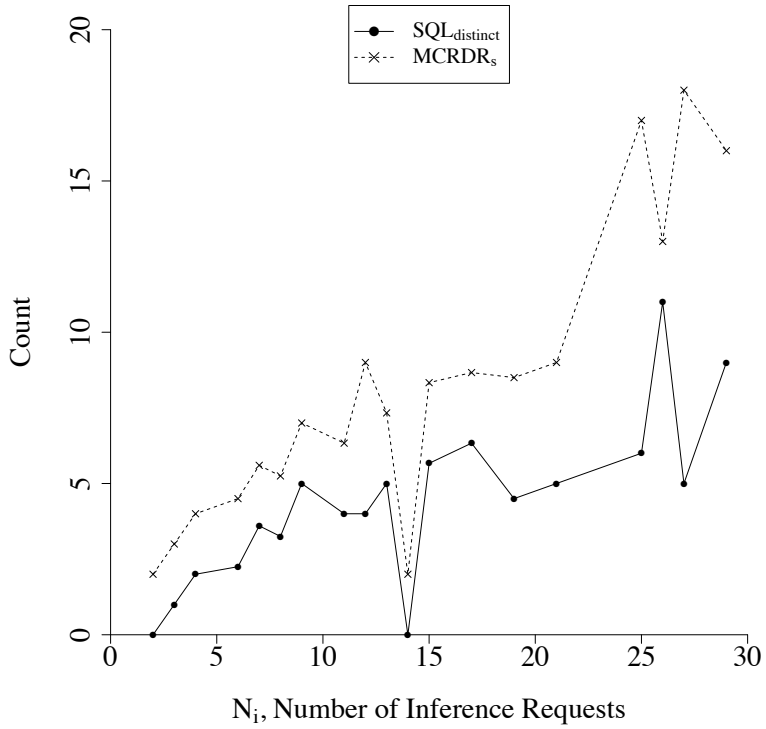


Figure 4.10: Count of distinct queries executed ($\text{SQL}_{distinct}$)

Figure 4.11 indicates the actual count of queries executed (SQL_{total}) based on the number of inference requests in a session, overlaid again with the equivalent number of standard MCRDR rules that would be required. Intuitively, assuming no repetition of the same request within a session, the number of queries executed in a session should approximate the same number of rules required in standard MCRDR, as in this domain, only two rules, R_0 (root) and R_1 (*hello*), of 36 do not result in post inference query execution. Figure 4.11 appears to validate this intuition. Here, in essence, multiple standard MCRDR rules have been replaced by the multiple execution of distinct queries bound by dynamically changing contextual data. The query count is also reduced in comparison to the standard MCRDR rule count due to the maintenance of conversational topic in the stack-based approach. The full list of queries defined for the C-MCRDR CA system can be found in Appendix A.2, Table A.9.

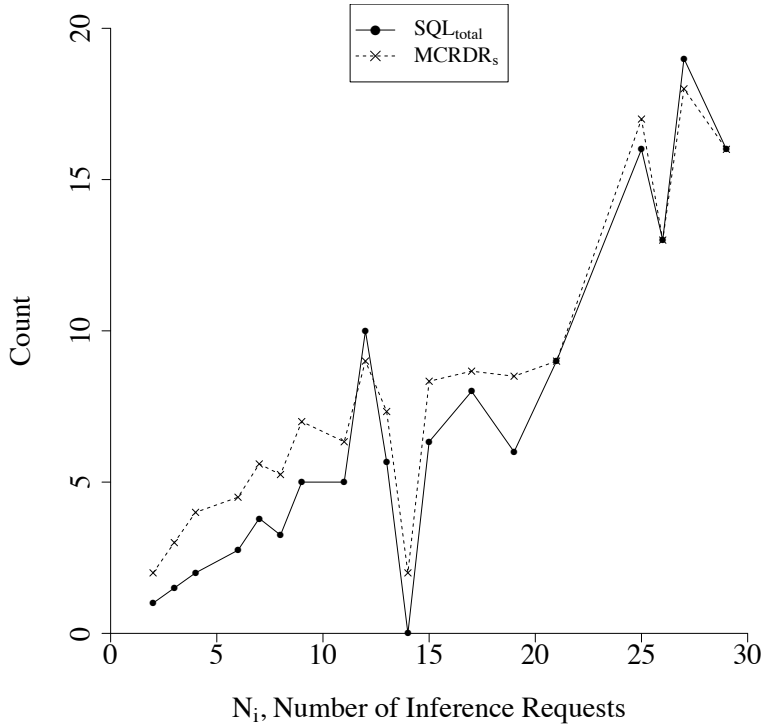


Figure 4.11: Count of all queries executed (SQL_{total})

4.2.6 Brittleness Mitigation

As discussed in Section 3.3.8, participants are prompted with examples of utterances that are understood from the current context, triggered if the last user utterance only satisfied the default root rule. Alternatively, a meta-rule that matches the utterance *What can I say?* has a consequence containing `METAHELP` (with the final system response output as specified in Chapter 3, Equation 3.3.8.1.4), which is also a trigger for brittleness mitigation. `METAHELP` was explicitly called 6 times in 4 distinct participation sessions out of the 41 participation sessions in total – if a user is unsure of what linguistic and domain coverage the system has, in this way they are given lookahead hints either explicitly when requested or as a result of a default rule response.

4.2.6.1 Default Rule Response Versus Non-default Rule Response

Evaluating the trigger rates for brittleness mitigation starts with Figure 4.12 where the raw system responses to inference request (irrespective of user intent and correctness of system reply) is shown.

Each participant session is ordered by the number of inference requests made in their session (N_i), and the stacked bar chart data shows the breakdown of the system response components by

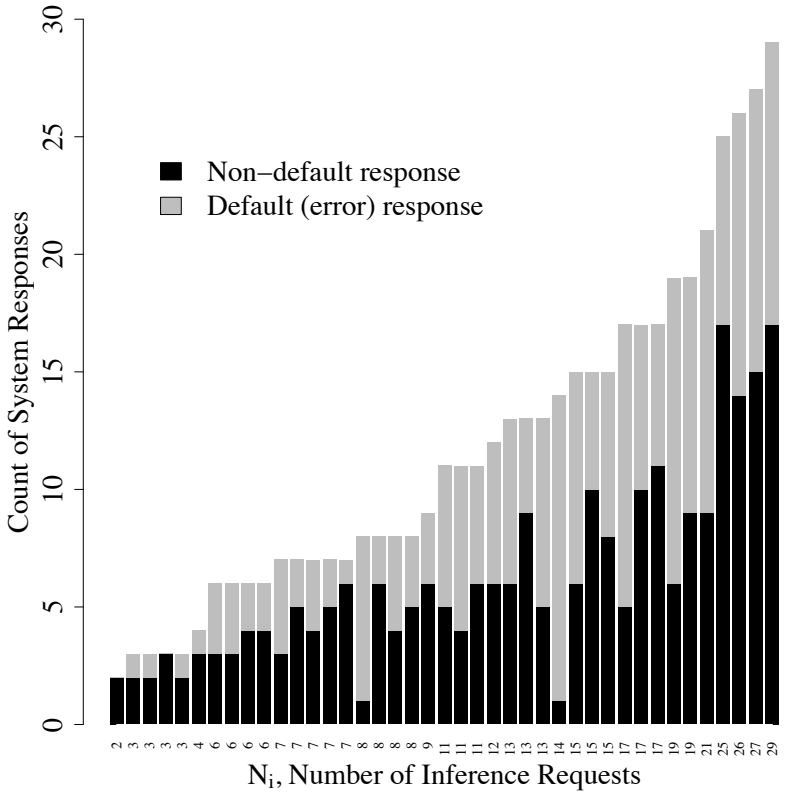


Figure 4.12: System response component breakdown

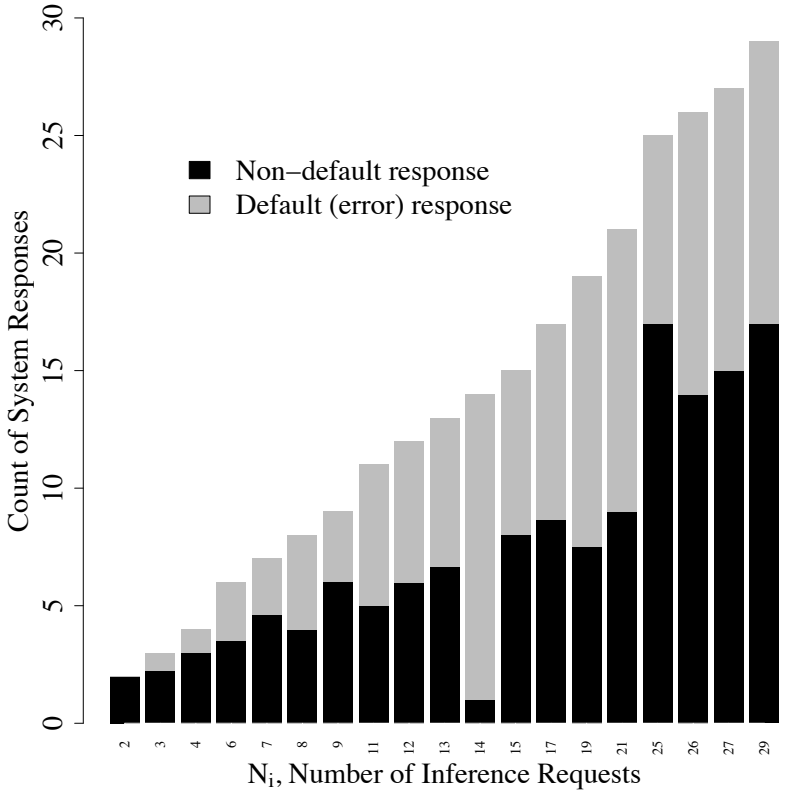


Figure 4.13: System response component breakdown (N_i averaged)

non-default responses (i.e. a result that does not satisfy the root rule) and default response (root rule is satisfied). Although the default root rule result is attributed to be an error (the system does not understand the utterance request), in many circumstances this is the appropriate (correct) result – this is investigated in the next section (Section 4.2.7) where the type of question and responses by categorising the results are investigated. Apart from two obvious outliers at $N_i = 8, 14$ (discussed in Section 4.2.8.1) the general trend of non-default responses increases by the number of inference requests made which is promising (and intuitive).

Aggregating the data (calculating the mean response counts for distinct N_i) produces the second figure in this section, Figure 4.13. There is still an outlier at $N_i = 14$ (as the source data only had one datum for this particular N_i and so averaging had no effect), but now linear trends seem apparent in both the default and non-default system response aggregate plots. Linear regression analysis determines there are significant linear models associated with the aggregate data, and this is demonstrated in Figures 4.14 and 4.15 for the non-default and default rule system responses respectively, noting the outlier at $N_i = 14$ has been omitted. The fitted models are:

$$C_{non} = 0.538N_i + 0.137, p = 3.46 \times 10^{-10}, R^2 = 0.91 \tag{4.2}$$

$$C_{def} = 0.461N_i - 0.137, p = 3.24 \times 10^{-9}, R^2 = 0.89 \tag{4.3}$$

Despite the linear trend models shown above, the stacked bars in Figure 4.13 portrays a false visualisation that the error responses have a steeper gradient than the non-default responses, and so Figure 4.16 shows the components side-by-side for direct comparison. In both figures the magnitude of the components (non-default rule versus default rule) is evident, and the actual ratios between default response component against non-default have been plotted in Figure 4.17 with the mean value ($\mu = 0.79$, $\sigma = 0.39$) represented by the dotted horizontal line (please note outlier $N_i = 14$ has also been omitted in this plot). Ratio values above 1 indicate where the session (averaged for distinct number of inference requests) has more default rule responses compared to non-default, and this occurs for $N_i = 11, 19, 21$. Inspection of the original log file data for the individual sessions associated with $N_i = 11, 19, 21$ show (noting 3 sessions had 11 inference requests, and 2 sessions had 19) the reasons for the high ratio values – these are shown in the following table, Table 4.5.

Irrespective of actual user intent and the *appropriateness* of the system response categorisation (which will mitigate default rule responses), a mean ratio of 0.79 is very encouraging as this shows the non-default rule system response occurs (on average) more than 1.27 times more often than the default rule response.

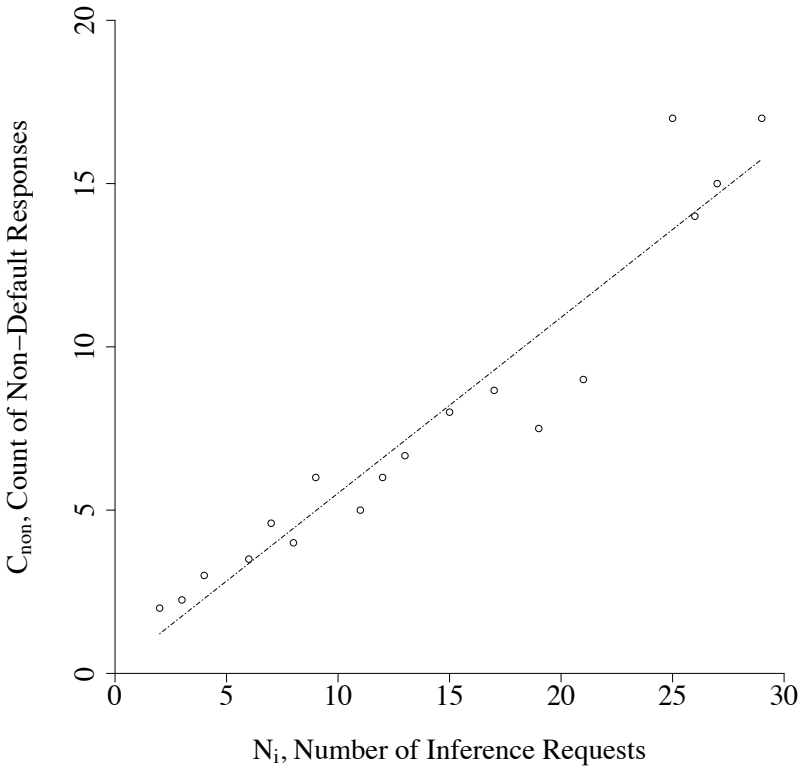


Figure 4.14: Count of non-default responses

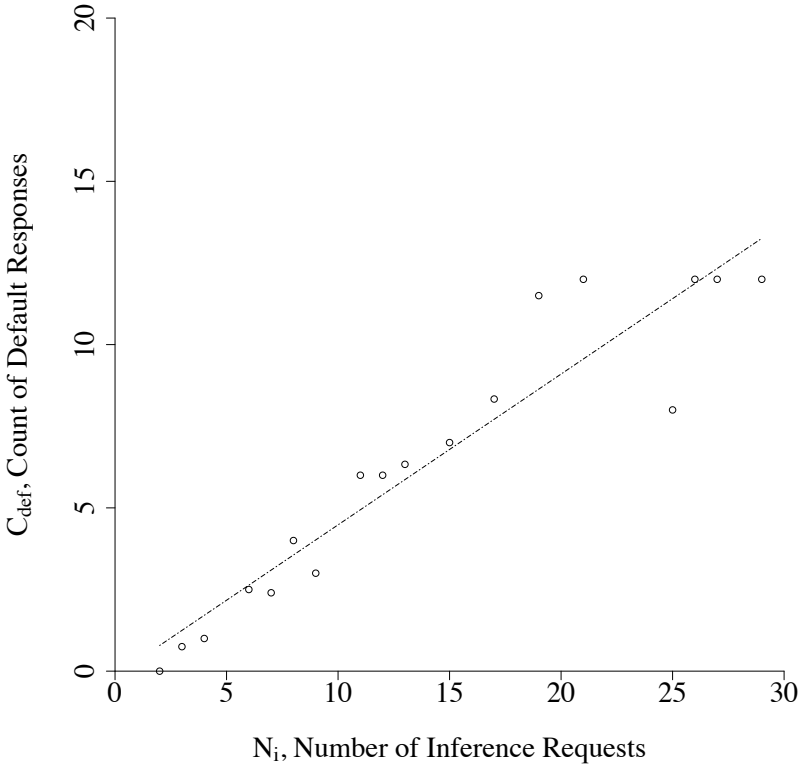


Figure 4.15: Count of default responses

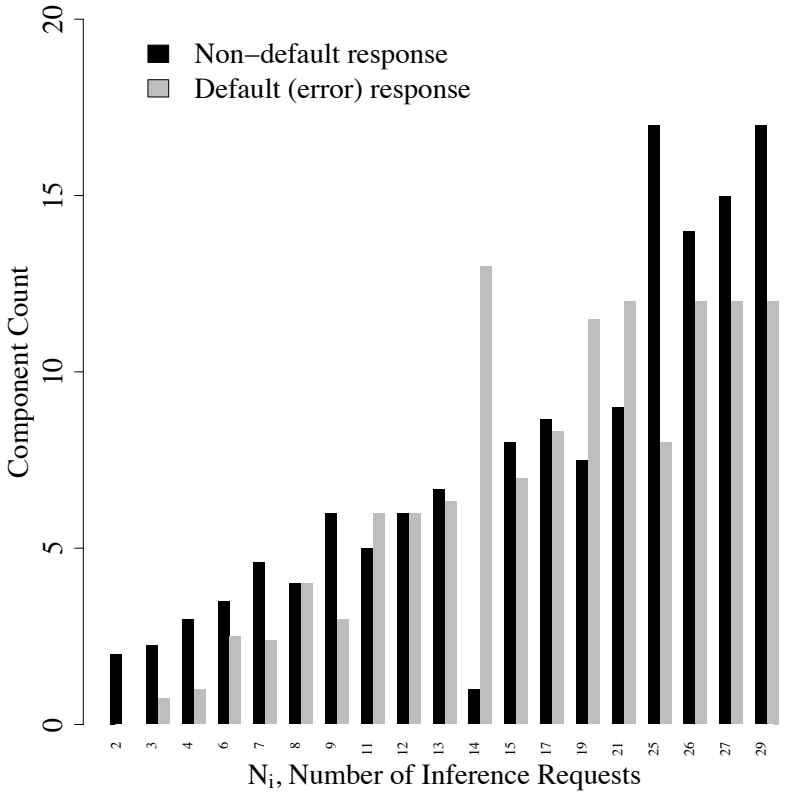


Figure 4.16: System response component comparison (N_i averaged)

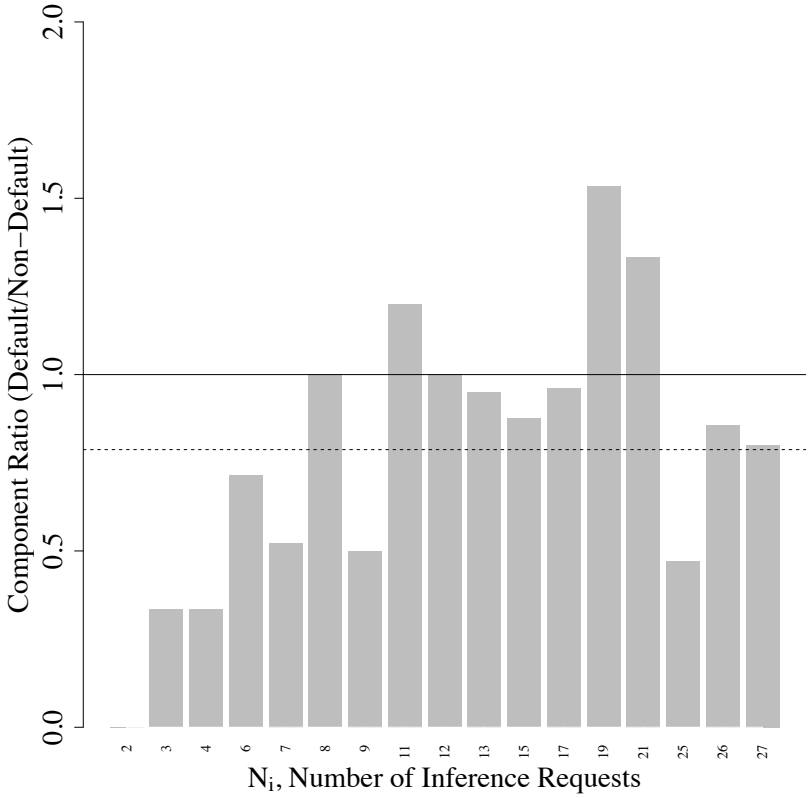


Figure 4.17: Component ratio (N_i averaged), $\mu = 0.79, \sigma = 0.39$

Table 4.5: High default rule count (DRC) to non default rule count ratio reasons

N_i	Sess	Reason	DRC	Comment
11	1	out-of-scope	6	e.g. timetable queries and *where is the uni bar?*
11	2	out-of-scope	5	timetable queries
		non-sensical/misspelled	2	
11	3	incorrect context	1	requesting assignment specific item without prior specifying an assignment
		non-sensical/misspelled	4	
19	1	out-of-scope	5	timetable queries and requests for units not offered
		non-sensical/misspelled	8	
19	2	incorrect context	2	e.g. asking for assignment due date without prior specifying an assignment
		non-sensical/misspelled	9	e.g. *Siri is cooler than you*
21	1	out-of-scope	6	e.g. request for tutor names, or *is it hard?*
		non-sensical/misspelled	6	

4.2.6.2 Brittleness Mitigation Component Breakdown

The effect of brittleness mitigation is examined by investigating the component contribution of non-default rule system responses that occurred due to prior prompting, in effect, an inference response that resulted in a default rule response that was also coupled with a lookahead hint (as defined in Equation 3.32 in Chapter 3). Manual inspection of original participant sessional logs determined where an inference request resulted in a non-default rule system response, and identified whether it was directly (with high probability) associated with a previously hinted term given by the term replacement function R_T defined in Equation 3.30 in Section 3.3.8.1.1. In other words, the participant has seen a lookahead hint in reply to a default rule response (*I don't understand*) and has then used one of the hinted paraphrasal terms in a followup inference request. Please note to avoid excessively long hint responses, the maximum number of random synonym (R_T) terms provided in response for a paraphrasal hint is limited to 5.

Figures 4.18 and 4.19 display the same data as previous Figures 4.12 and 4.13, except the contribution of brittleness mitigation counts can now be seen (the red coloured bars), for all of the original participant session data (Figure 4.18), and for the participant data averaged by distinct N_i (Figure 4.19). In both figures, the default rule responses are still shown in light-grey, but now the non-default responses are broken into their two sub-components: responses that are as a result of spontaneous (previously unprompted) inference requests (shown in black),

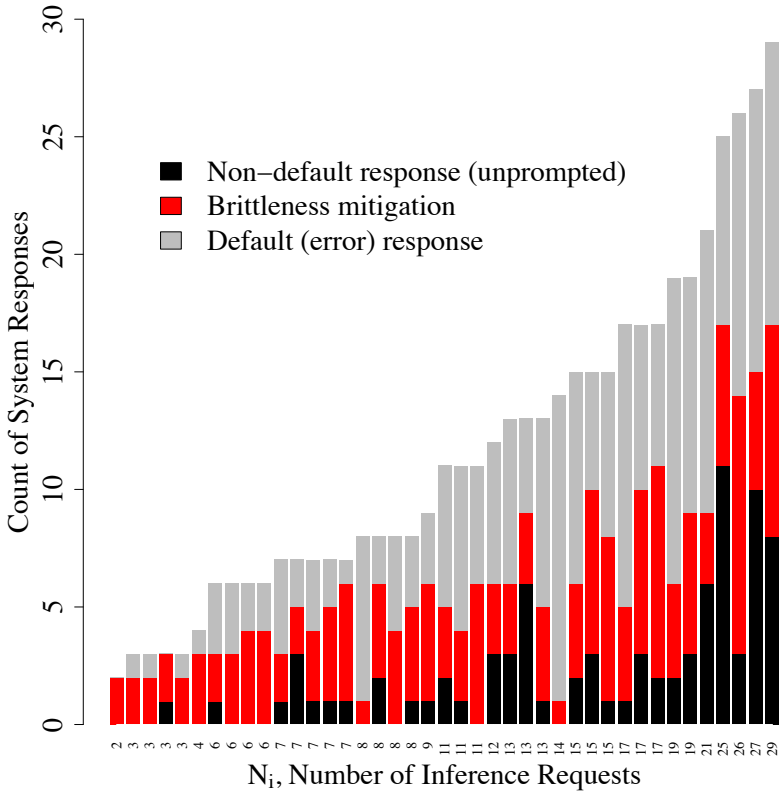


Figure 4.18: System response full component comparison

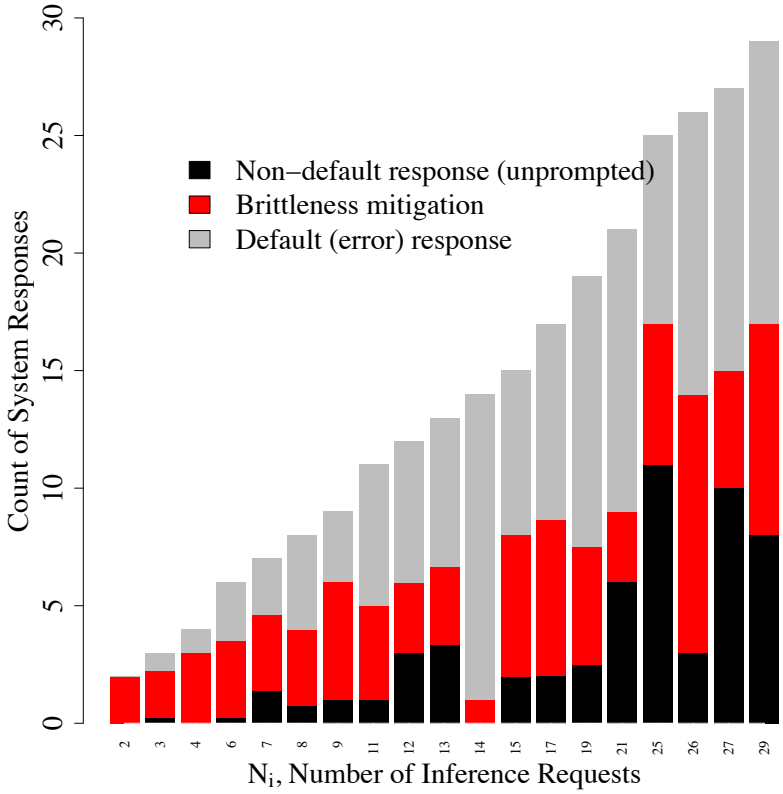


Figure 4.19: System response full component comparison (N_i averaged)

and responses that are highly likely the result of inference requests that have been influenced by previous brittleness mitigation hints (shown in red). Both figures from visual inspection show the brittleness mitigation component of non-default rule system replies is typically more than 60–70% of the combined non-default responses in each case. This observation is confirmed through Figure 4.20 which plots the ratios of the brittleness mitigation component against the total number of non-default rule system responses for each N_i (again, averaged for each distinct N_i). General observations that can be gleaned here are that brittleness mitigation is extremely important, especially, for example, sessions with few inference requests i.e. $N_i = 2, 3, 4, 6$, as almost every response is the result of the system providing a prior lookahead hint. High N_i, for example $N_i = 21, 25, 27, 29$, show participants adopting appropriate vocabulary without the need for excessive prompting (as only approximately 30–50% of prompting was needed).

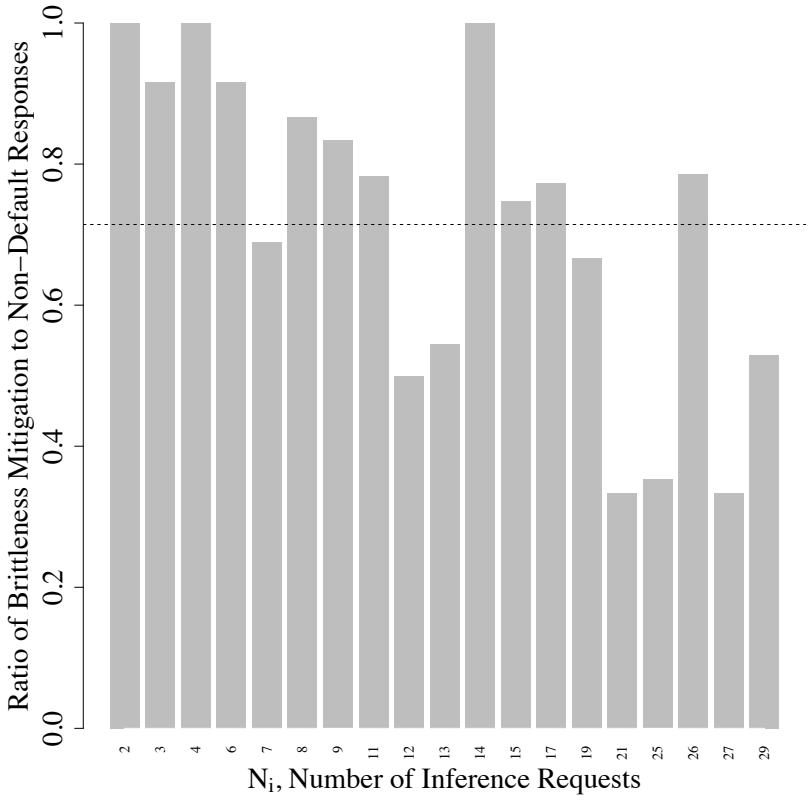


Figure 4.20: Ratio of brittleness mitigation to non-default responses (N_i averaged), $\mu = 0.71$,$\sigma = 0.22$

4.2.6.3 Brittleness Mitigation Summary

When the the ratio of default rule response counts by non-default rule response counts for participant sessions is investigated, a mean ratio of 0.79 ($\sigma = 0.39$) is found, or in other words, non-default rule responses occur on average 1.27 times more often than the default rule (*I don't understand*) response (irrespective of whether the default rule response is appropriate or not).

The evaluation results for the counts of non-default and default rule responses by the number of inference requests show significant linearity, with non-default responses growing by 0.538 for each inference request, and default responses having a smaller gradient of increase, 0.461 per inference request. The original log file data provides insight into reasons behind high default to non-default rule response ratios, and in particular the major reason appears to be out of scope questions (primarily related to timetable information) – this can be factored into future evaluation either via rule maintenance to include typical out-of-scope queries, or through partial brittleness mitigation indicating out-of-scope categories before participant sessions commence.

When the effect of brittleness mitigation is assessed, the mean ratio of comparison between (probable) mitigated inference requests and spontaneous (non-mitigated) inference requests (in both cases that resulted in non-default rule responses) is 0.71 ($\sigma = 0.22$). This result means (on average) that 71% of all inference queries that produced a non-default rule response were most likely due to brittleness mitigation in the form of paraphrasal lookahead hints from the current conversation context in each session (this ratio falls to the minimum value of 0.33 for specific sessions with that have 21 or 27 inference requests, and it has maximum value (1.0) for sessions with 2, 4 or 14 inference requests). Although the effect of evaluating system performance *without* paraphrasal lookahead hinting was not conducted, mitigation would appear to be extremely important from the participant's viewpoint (future evaluation may include examining the contribution (or lack thereof) of brittleness mitigation to system performance).

4.2.7 Question and Response Categorisation

After all sessional data was obtained, log files were then manually processed to categorise the system's responses to inference requests – for example, was the participant's utterance valid, did the system capture the correct intent, etcetera. This lead to the definition of four categories of *questions* and *responses* (see Table 4.6), that were applied in the discussion of results that follow.

Table 4.6: Inference result categories (C)

Category	Inference source and response
Cat1	Valid question, valid system response
Cat2	Valid question, misinterpreted system response
Cat3	Valid question, invalid (default) system response
Cat4	Invalid question, default system response

Referring to Table 4.6, observations include:

- **Cat1** is the "best" category as it indicates successful system responses to valid questions.
- **Cat2** can be problematic and responses of this type indicate additional rules need to be added to correct the response – key terms indicating the user's actual intent have been ignored.
- **Cat3** is the worst category as responses of this type indicate the system is unable to respond to a valid question (meaning it simply lacks an appropriate rule). The question however may have been completely out of scope, but from the user's perspective they have asked a valid question.
- **Cat4** responses indicate non-sensical user questions and thus the default response (*I don't understand*) is entirely valid.

The question and response categories defined in Table 4.6 are then applied to each of the sessional results in order to determine the effectiveness of system response (in the next section, Section 4.2.8), as well as gain insight into user acceptance of performance and response through the feedback system (Section 4.2.10).

4.2.8 System Performance

Inspection of the logged satisfied rule frequency results shows that 43% of inference requests (which result in Rule 0 responses, categories Cat3 and Cat4) did not produce a useful result for the user, while responses that satisfied Rules 1 - 36 (categories Cat1 and Cat2) had a combined frequency of 57%. However, this does not consider when Rule 0 responses are appropriate, or when Rules 1 - 36 responses are inappropriate, so two overall categories are considered – *appropriate system responses* and *inappropriate system responses* (defined in Definitions 4.4 and 4.5). Referring to Table 4.7, all inference requests (N_i) are grouped into their Cat1 - Cat4 categories and expressed as a percentage of the total of all inference requests. The system's performance in terms of the *appropriateness* of system response can be defined and summarised

as:

Appropriate responses: combine categories Cat1 and Cat4 (4.4)

Inappropriate responses: combine categories Cat2 and Cat3 (4.5)

1. The system is responding **appropriately**: **80.3%** of all responses
2. The system has **inappropriate** responses: **19.7%** of all responses

Table 4.7: Categorised system responses

Category	$\mathbf{N}_i$	**% Total**
Cat1	252	52.72
Cat2	21	4.39
Cat3	73	15.27
Cat4	132	27.62
Total	**478**	**100**

N_i = number of inference requests

In Section 4.2.4, Figure 4.3 was also plotted as a column graph (Figure 4.4) to highlight the differentiation between the default rule response frequency and all other rules. This indicated approximately 43% of inference requests (which now include Cat3 and Cat4 responses) did not produce a useful result for the user. Using the categorisation criteria defined in Table 4.7, an alternative view to Figure 4.4 is Figure 4.21 where it now considers that the appropriate system response to non-sensical (invalid) questions is the *question is not understood*. Figure 4.21 provides a useful visual breakdown of the categorical data by alignment to the *appropriateness* of the system response. The overall *appropriate* system response rate of 80.3% is very promising considering the fact that the developed C-MCRDR knowledge-base is a minimal encoding of the unit outline domain. Improvement can be achieved by reducing the category Cat2 and Cat3 rates – these types of responses can be corrected by the addition of new rules which in C-MCRDR terms is a refinement of the final classifications, however during evaluation this was not conducted; Cat2 responses indicates key terms in the utterance (the actual intent) have been ignored or misinterpreted, whereas Cat3 responses indicates the system is unable to respond to a valid (but out-of-scope) question. The category Cat4 rates, which are appropriate responses to non-sensical input, are difficult to reduce without more intensive user training, further brittleness mitigation, or constraining input mechanisms to text-only (as the largest contributor to this category came from ASR errors - see Section 4.2.9). Further rule and

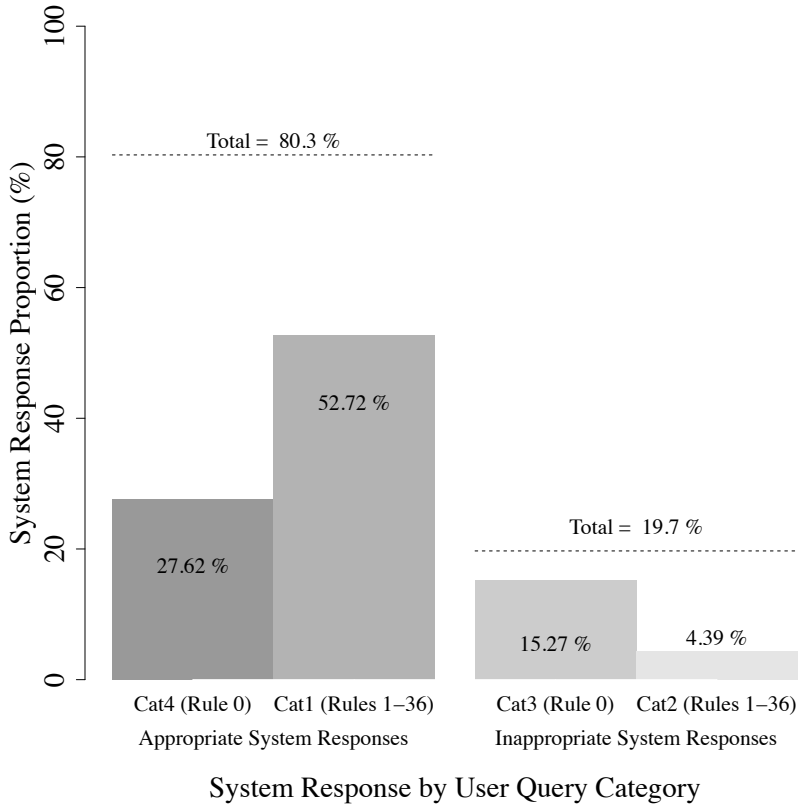


Figure 4.21: Overall system response rates

dictionary maintenance would improve the overall results and this will be considered in future work – please see Section 5.3, however a brief indicative speculative analysis of the logged inference session data to determine user intent follows:

4.2.8.0.1 Cat2 Mitigation by Rule/Dictionary Maintenance

For the Cat2 (misinterpreted) responses ($N_i = 21$):

- eight responses can be simply corrected by paraphrasal (dictionary) maintenance (for seven terms);
- eleven responses can be corrected by three additional rules (two associated with timetabling) and the third associated with out-of-scope unit codes – these rules will also mitigate Cat3 responses as detailed next;
- two responses require two new rules for special cases: e.g. "*besides X and Y, what other engagement activities are there?*" and "*should I do unit X or unit Y?*"

4.2.8.0.2 Cat3 Mitigation by Rule/Dictionary Maintenance

For the Cat3 (default response) responses ($N_i = 73$):

- 41 responses can be simply corrected by paraphrasal (dictionary) maintenance (for 23 terms);
- 31 responses can be corrected by 8 additional rules that would be comprised of timetabling information, venue information (from timetabling), out-of-scope unit codes, and sessional teaching staff (tutors), also from timetabling.
- 19 out of the 31 responses from above (61.3%) are associated with timetabling-related questions. This indicates a considerable portion of the out-of-scope requests can be mitigated by the addition of three rules (one for class times, one for class venues and one for tutors associated with class times).

4.2.8.1 Cat1 Responses

Analysing system performance but isolated to Cat1 responses yields Figure 4.22. The *Ideal* response rate is shown, $[NCat1 = N_i]$ as is the *Best fit* linear regression model ($R^2 = 0.93, p < 2.2 \times 10^{-16}$), $[NCat1 = 0.52 \times N_i]$. This model shows fundamentally, across all participation sessions, 52% of inference requests are valid and receive a non-default response (which agrees closely with the Cat1 % total in Table 4.7). There are two obvious outliers - one at $N_i = 8$, $NCat1 = 1$ and another at $N_i = 14$, $NCat1 = 1$. In both cases their primary input was via speech (75% and 78% of their requests respectively), and they failed to articulate (voice) a single unit code.

An alternative visualisation of Figure 4.22 is where the Cat1 responses are considered as a proportion of the total number of inference requests across all categories in each session (Figure 4.23). As can be seen, here the mean (μ) Cat1 response rate is 56.63% with standard deviation, $\sigma = 19.04\%$ (whereas the ideal rate is 100% if all responses were Cat1). The considerable deviation around the mean is also reflected by a very low R^2 value (0.10) when a linear regression model was fitted ($[RCat1 = 67.54 - 0.94 \times N_i]$, $R^2 = 0.10$, $p = 0.03$). The negative trend of the fitted model is heavily influenced again by the same significantly low outliers in sessions with 8 and 14 inference requests. The Cat1 response rate here (56.63%) differs to that in Table 4.7 as here Cat1 responses are averaged per-session, and a global mean response rate is calculated across all sessions.

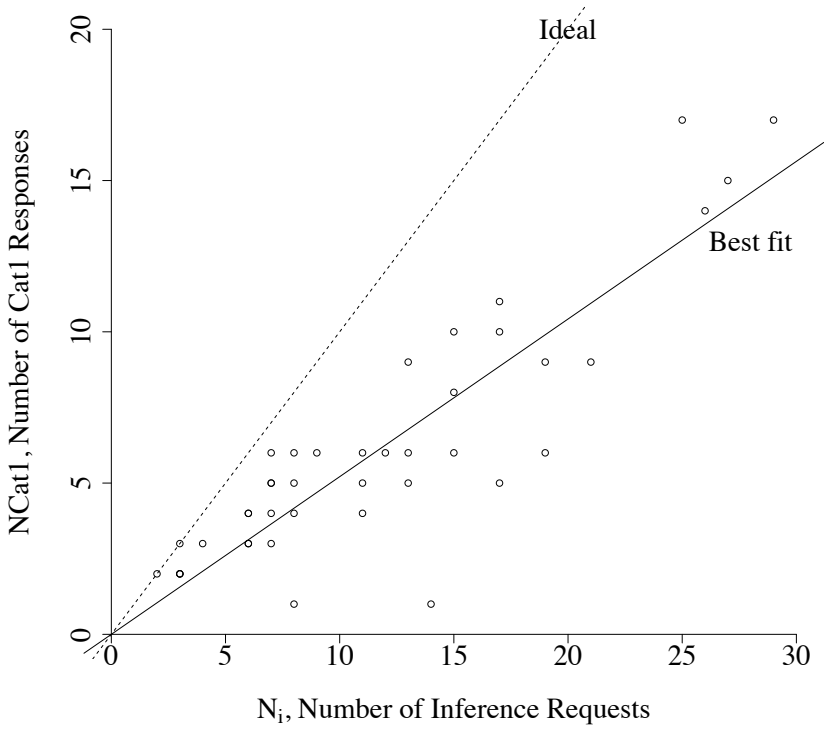


Figure 4.22: Cat1 response totals across inference sessions

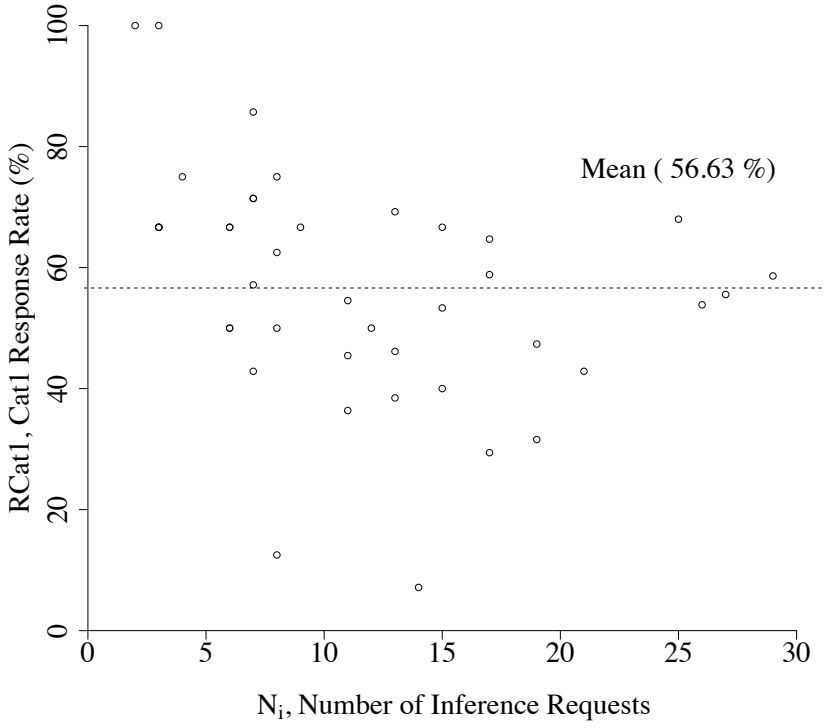


Figure 4.23: Cat1 response rate % across inference sessions
$\mu = 56.62\%$, $\sigma = 19.04\%$

4.2.8.2 Cat2 Responses

Figures 4.24 and 4.25 provide plots of the Cat2 response rates (initially with response totals, followed by totals as a proportion of the total number of inference requests across all categories in each session). Figure 4.24 has a statistically significant linear regression model line of best fit, $[NCat2 = 0.042 \times N_i]$,$(R^2 = 0.19, p < 2.39 \times 10^{-3})$.

Considering the response proportion totals (Figure 4.25), there is a large variability of the data, and no statistically significant linear regression model can be fitted ($R^2 = 0.001$, $p = 0.82$). It is however encouraging that such a low average misinterpretation rate is found in the evaluation.

4.2.8.3 Cat3 Responses

Figures 4.26 and 4.27 provide plots of the Cat3 response rates. Figure 4.26 has a statistically significant linear regression model line of best fit, $[NCat3 = 0.15 \times N_i]$,$(R^2 = 0.54, p < 1.59 \times 10^{-8})$.

No statistically significant linear regression model can be fitted ($R^2 = 0.012$, $p = 0.50$) when considering the response proportion totals (Figure 4.27). A positive outcome here is that the mean is low (14.38%), meaning the worst system performance measure in the domain is not excessive and it is indicative that rule coverage of the domain (in the evaluation) is relatively high. There is a large outlier ($N_i = 20$, RCat3=57.89%) indicating nearly 58% of responses (11 responses, which can also be seen in Figure 4.26) for a session with 19 inference requests produced invalid responses to seemingly valid questions. Examining the logged conversation data indicated the participant repeatedly asked for unknown unit codes.

4.2.8.4 Cat4 Responses

Figures 4.28 and 4.29 show the plots of the Cat4 response rates. Figure 4.28 has a statistically significant linear regression model line of best fit, $[NCat4 = 0.29 \times N_i]$,$(R^2 = 0.71, p < 1.57 \times 10^{-12})$.

No statistically significant linear regression model can be fitted ($R^2 = 0.034$, $p = 0.12$) for the data shown in Figure 4.29. It is worth noting here that every non-zero RCat4 response is due to an invalid (or nonsensical) question. An example outlier, ($N_i = 8$, RCat4=87.50%) has 7 default-rule responses as a result of invalid questions (inarticulate unit codes), however, this can be linked to ASR errors as the participant's primary input was via speech (see Figures 4.30 and 4.31 in Section 4.2.9). The category Cat4 rates, while still appropriate as responses to nonsensical input, are difficult to reduce without more intensive user training, further brittleness

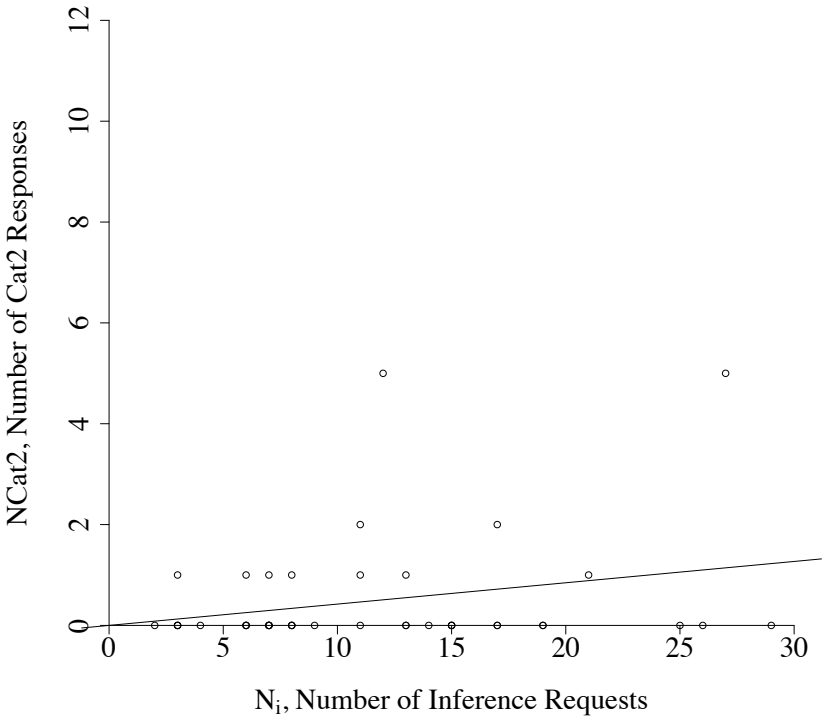


Figure 4.24: Cat2 response totals across inference sessions

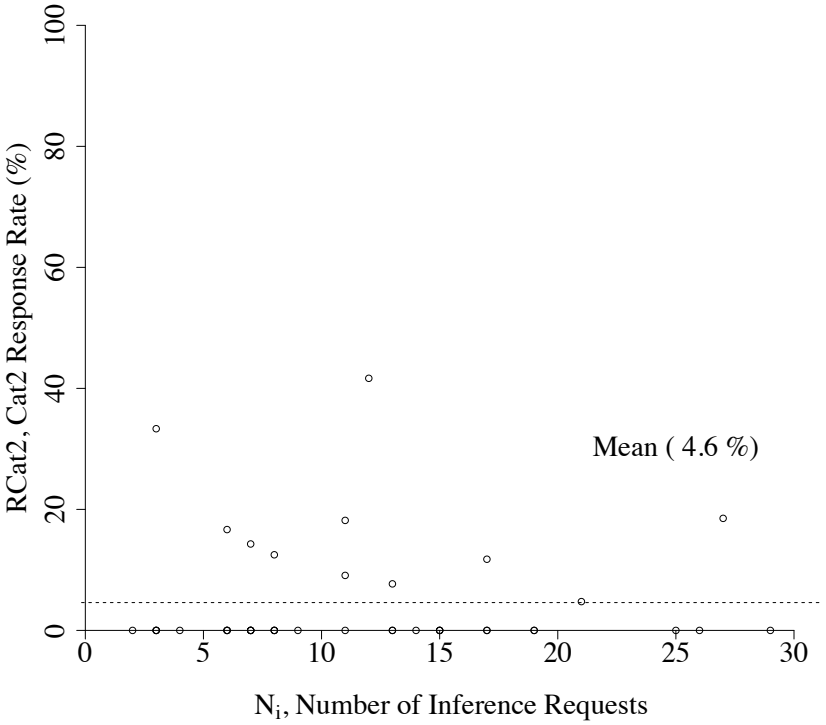


Figure 4.25: Cat2 response rate % across inference sessions
$\mu = 4.60\%$, $\sigma = 9.48\%$

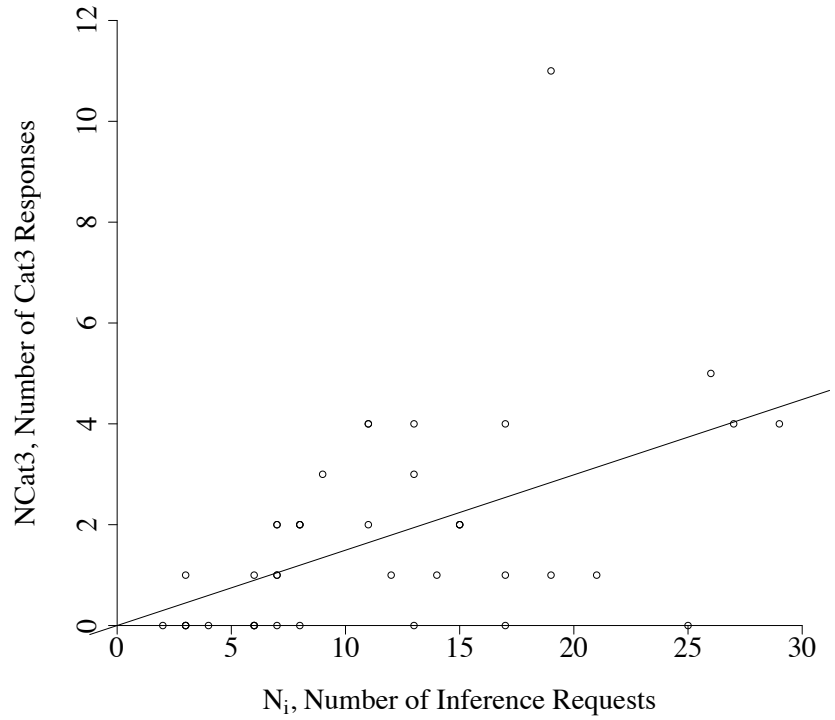


Figure 4.26: Cat3 response totals across inference sessions

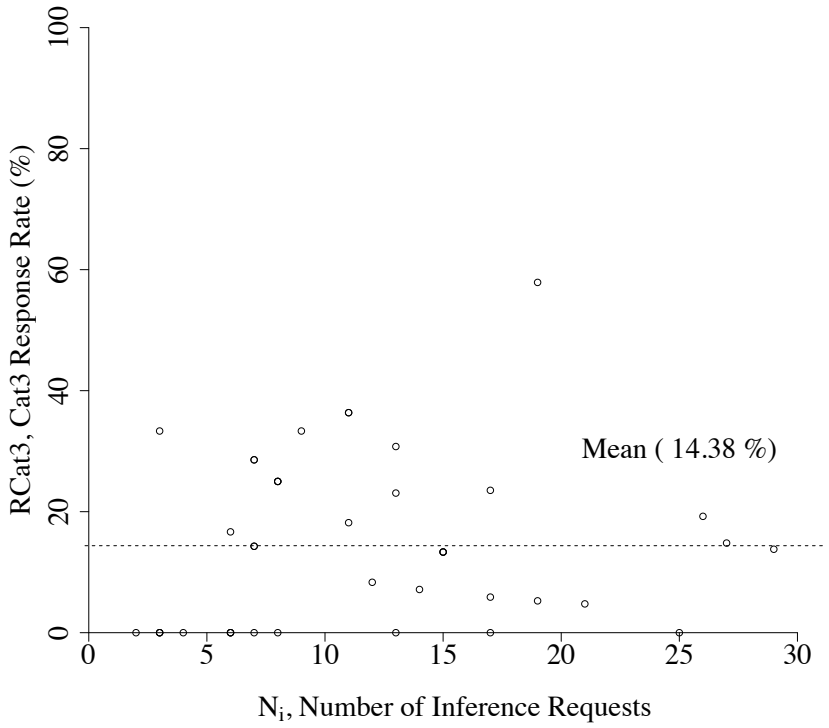


Figure 4.27: Cat3 response rate % across inference sessions
$\mu = 14.38\%$, $\sigma = 13.98\%$

mitigation, or constraining input mechanisms to text-only (as the largest contributor to this category came from ASR errors).

4.2.8.5 System Performance Summary

By grouping system response data by category means the system responses can be aligned with the participant's competency and intent when they formulate questions for inference processing. This gives a more insightful view of the system performance when considering these factors. In particular, the *appropriateness* of the system response can be defined by considering the combination of responses that are categorised by the type of participant question and the actual system response. As has been mentioned in Section 4.2.8, it was established that the C-MCRDR CA system, with a relatively shallow knowledge-base, had achieved an *appropriate* system response rate of 80.3%.

Table 4.8: Category utterance examples

Cat	**Utterance**	**Rules**	**Comment**
Cat1	*what is the assessment pattern?*	R1-36	Valid question and response
Cat1	*what are the prerequisites?*	R1-36	Valid question and response
Cat2	*what computer labs can I use?*	R1-36	Misinterpreted, out-of-scope; "labs" is a synonym for "*classes*" and the response summarised the types of classes in the unit. The actual intent was to determine which computer rooms were available (timetabling)
Cat2	*KIT001 tutorial times?*	R1-36	Misinterpreted, out-of-scope; *KIT001* matches a unit code (Rule 3), and *tutorial times* was ignored. Timetable schedules were out-of-scope but were frequently asked for.
Cat3	*who is the tutor?*	R0	Valid question, but out-of-scope; Tutor allocation requires timetabling data
Cat3	*what work can KIT001 lead to?*	R0	Valid question – some analysis and inclusion of more synonyms such as *lead to* for the dictionary term *learningoutcomes* may have facilitated this question
Cat4	*when cold idle*	R0	Invalid question
Cat4	*what about Co 80205*	R0	Invalid question

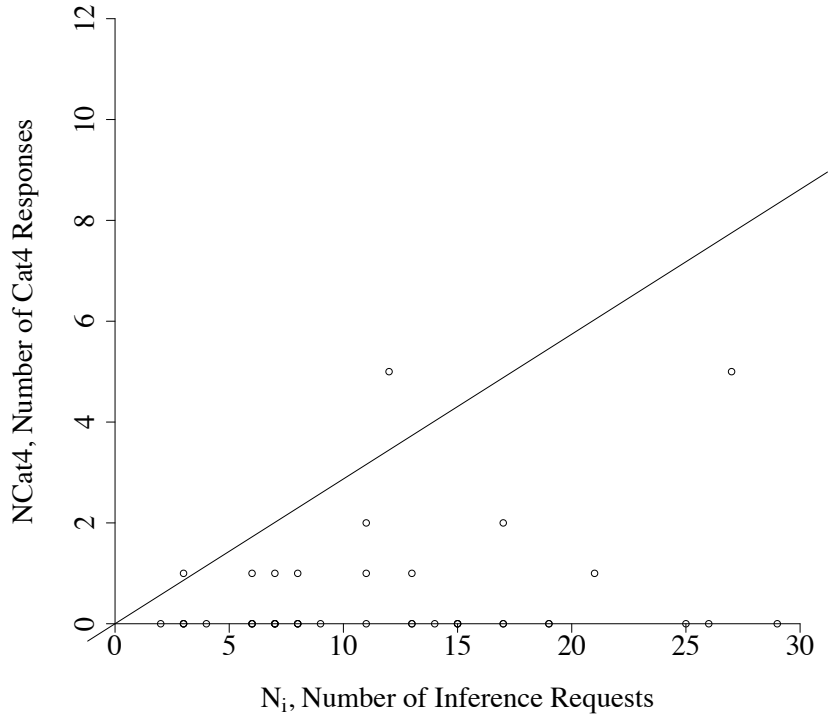


Figure 4.28: Cat4 response totals across inference sessions

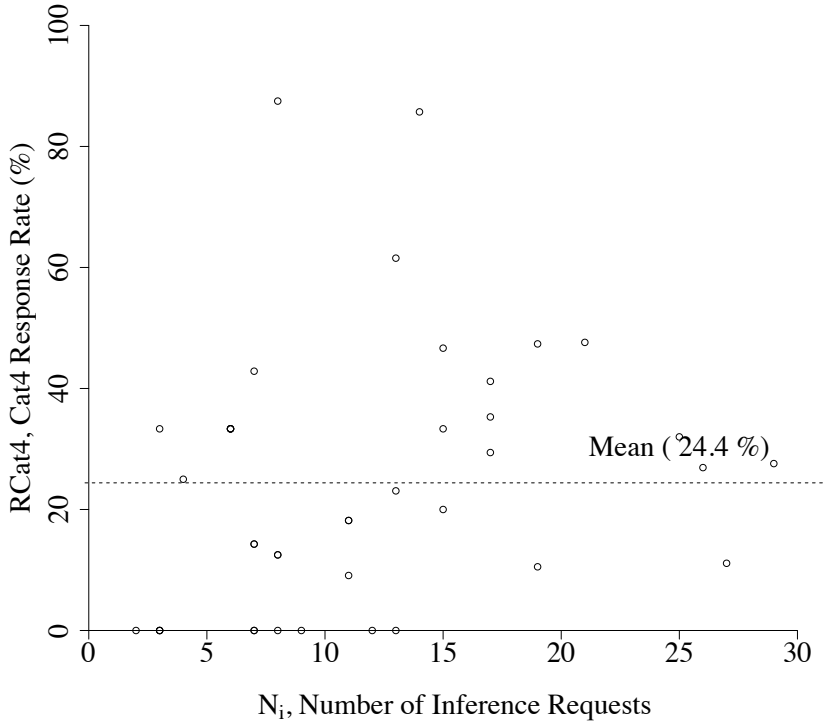


Figure 4.29: Cat4 response rate % across inference sessions
$\mu = 24.40\%$, $\sigma = 21.88\%$

4.2.9 Automatic Speech Recognition (ASR) errors

Results from this section help to answer research question SRQ3 (see Section 1.2.1).

4.2.9.1 Speech-to-text Correction

The system implementation includes a speech-to-text capability via the Chrome browser's embedded Google Speech API (Section 3.3.9.1 (see Figure 3.15 in Section 3.3.11), (Google, 2018*b*)), and so common ASR corrections encountered during the author's construction of the unit outline knowledge-base included replacing one-to-one terms, for example, *unicorn* with *unitcode* and correcting individual unit code identifiers (which are difficult to correctly recognise as they are codes, not standard words). A unit code consists of a three letter code followed by three digits (for example, *KIT001* or *KIT304*), and it is used directly as querying criteria against an *in situ* database table with unit code as a primary key. Example regular expressions for correction are shown in Table 4.9 and in total, 55 speech-to-text corrections were defined for the domain (see Table A.1 in Appendix A.1).

Table 4.9: Regular expressions for correction

Rule	**Replacement**	**Example**
`kicks (/d)`	`KIT$1`	*kicks 123→KIT123*
`unicorn`	`unitcode`	*unicorn→unitcode*
`item for`	`item 4`	*item for→ item 4*
`one`	`1`	*one→ 1*
`.*[i][t] (/d)`	`KIT$1`	*crit 123→KIT123*

4.2.9.2 Text-to-speech Correction

Although not extensively used by participants in the C-MCRDR CA system evaluation, 66 text-to-speech corrections were added, primarily correcting speech pronunciation of acronyms, times and proper nouns; corrections were also added due to database-specific values retrieved that do not translate directly from textual to spoken form naturally. The complete list of corrections can be found in Table A.2 in Appendix A.1.

4.2.9.3 Evaluation ASR Errors

The speech input and output components of the system were not fully utilised or embraced in this domain. This is not surprising as most evaluation was conducted in noisy laboratory-based

environments which made spoken questions and responses harder to correctly recognise. There was a scarcity of data logged with only 11 sessions (i.e. 26.8% of sessions) using speech – this is shown in Figures 4.30 and 4.31 and summarised in Table 4.10. The *ASR Error Rate* is the mean percentage of ASR transcription errors that occurred across all session requests that used speech, and the *ASR Duration* is the average measure of when speech was used in a session, how many requests were actually conducted by speech.

The high standard deviations in Table 4.10 are indicative that particular (outlier) users had greater success with speech input. For example, a session at $N_i = 27$ used ASR for 90% of their session, yet only suffered 11% ASR errors. In contrast, a session at $N_i = 14$ used ASR for 79% of their session duration, and suffered 64% ASR errors. The majority of participation sessions immediately modified the input mechanism from speech to text, but for those sessions that started with speech, 72.7% persevered with it for an average of 75.91% of the session requests. ASR transcription errors are mitigated for common mistakes (for example, correcting *unitcode* utterances) via rule-based corrections as detailed in Sections 3.4.2 and 4.2.9.1. Note however during evaluation, no additional corrective rules were added beyond the initial set to the knowledge-base. Improvement on the ASR error rate by the adoption of best-performing market-leading IPA device is evaluated in Section 4.3.

Table 4.10: ASR data $|N_i| = 11$

Statistic	μ	σ
ASR Error Rate	26.14%	23.20%
ASR Duration	75.91%	35.18%

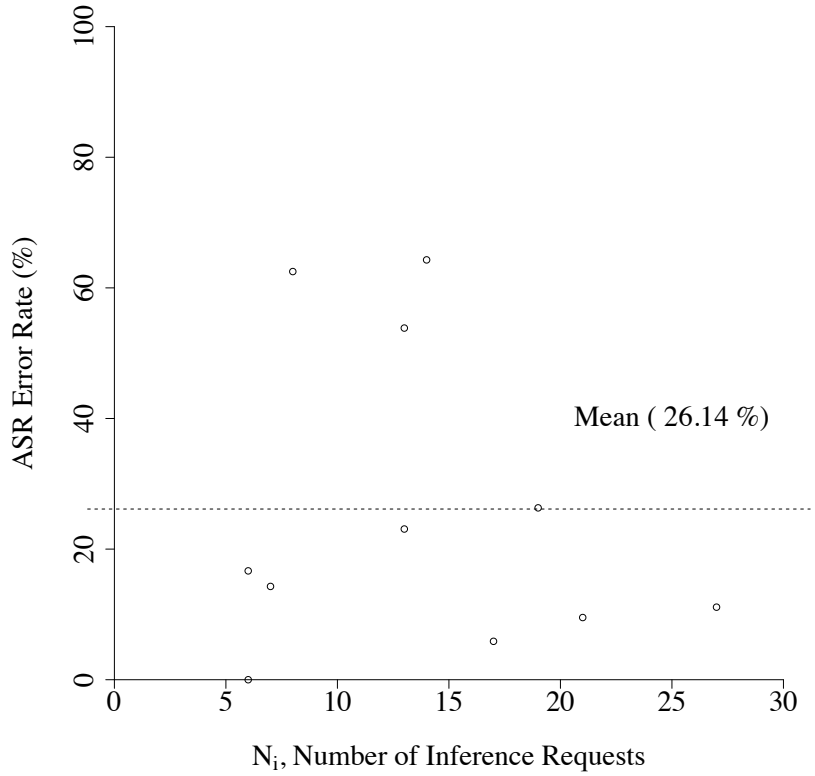


Figure 4.30: ASR Error % across inference sessions

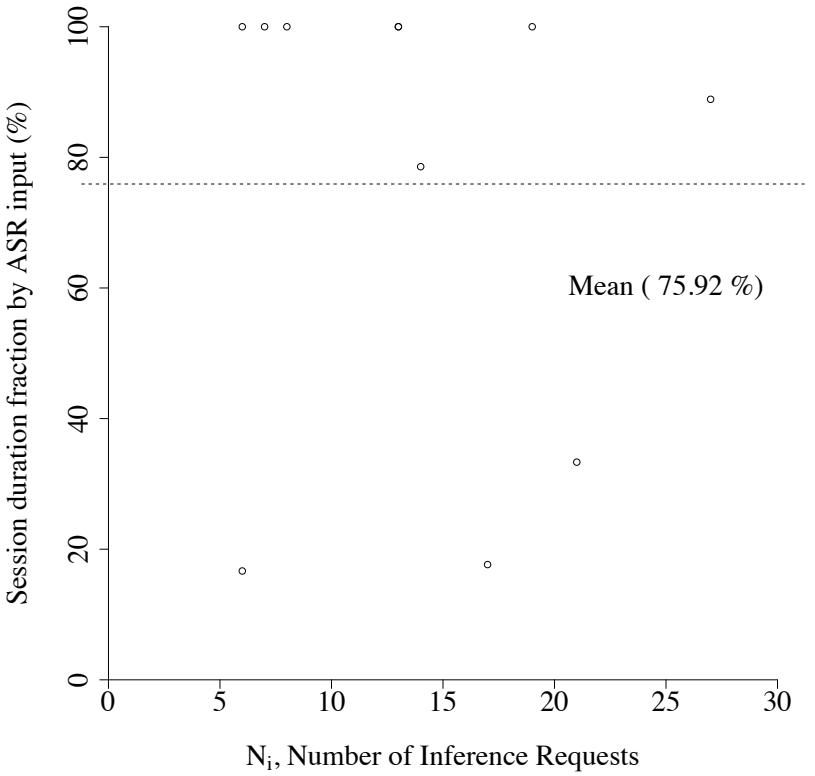


Figure 4.31: ASR session duration % across inference sessions

4.2.10 User Acceptance (Feedback)

Results from this section help to answer research question SRQ3 (see Section 1.2.1).

User acceptance was evaluated by two aspects of voluntary feedback (see Section 3.3.13.1):

(a) the usability of the entire system (a qualitative perceptive evaluation of the overall system performance); and

(b) individual system responses to inference requests.

The Likert scales were defined as follows:
(a) Rating overall system usability:

1. *I am very dissatisfied*
2. *I am slightly dissatisfied*
3. *I am neither satisfied nor dissatisfied*
4. *I am slightly satisfied*
5. *I am very satisfied*

(b) Rating the system responses to each question submitted:

1. *My answer is totally wrong*
2. *My answer is partially wrong*
3. *My answer is neither right nor wrong*
4. *My answer is correct but some information is missing*
5. *My answer is exactly what you are seeking*

Although not direct feedback, participants can send their dialog chat history to the coordinating lecturer for clarification of unanswered (or mis-answered) questions as detailed in Section 3.3.8. Interestingly, no participant chose this option. In total, 41 sessions by undergraduate volunteers in the ICT Discipline were logged. Although this is a relatively small sample size, it is larger and comparable with other similar satisfaction studies (Smith et al., 2014; Shabaz et al., 2015). The sessional data was grouped based on the number of inference requests that were made, and where relevant, participant response ratings were averaged per group.

The individual feedback responses were then considered against the individual rules that were satisfied, and against the category of response introduced in Section 4.2.7.

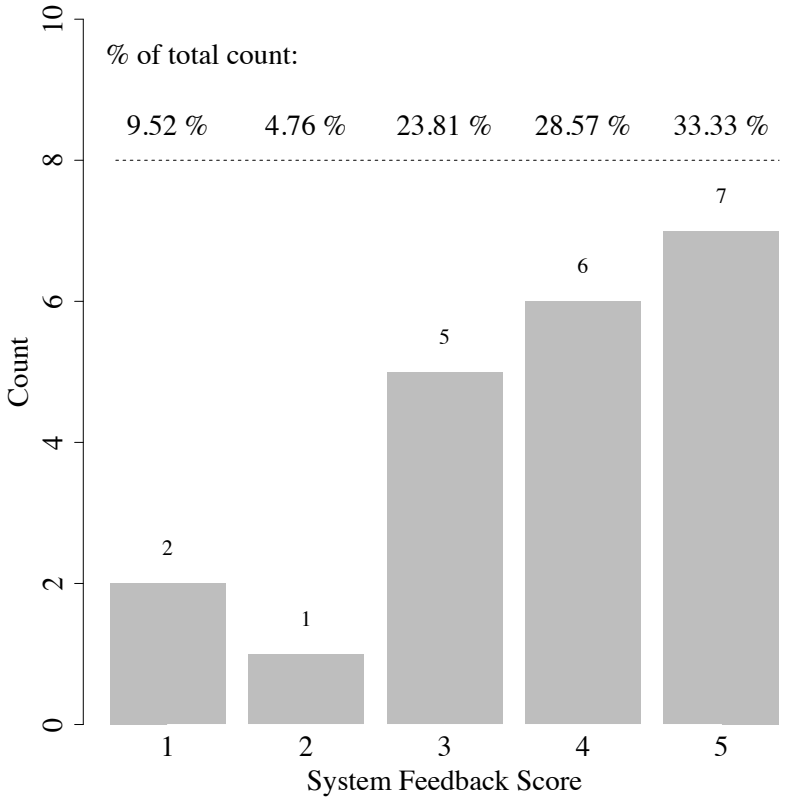


Figure 4.32: Count of system feedback scores ($|N_i| = 21$)

4.2.10.1 System Satisfaction Feedback

Users could optionally rate their satisfaction of the entire system, and 51.2% of sessions ($|N_i| = 21$) did so. Overall feedback was very positive - the highest frequency scores (on a 5-point Likert scale, 1=*I am very dissatisfied*, 5=*I am very satisfied*) were ratings at levels 4 and 5 (28.57% and 33.33% respectively), and there was a combined (levels 1 and 2) negative rating of 14.28% – see Figure 4.32. The breakdown of per-session system feedback scores are summarised in Figure 4.33. Overall, only 3 users rated the system below the neutral rating of 3 (shown as the darker coloured bars), and it may be of interest to ascertain why. As users could also provide an explanatory comment, the 3 user's comments were as follows:

- user 14 inference requests – rating:1, *doesnt recognise my voice at all correctly.response voice is too quick , too fast paced*
- one user with 15 inference requests – rating:1, *it require a further learning*(sic)

- user at 29 inference requests – rating:2, *Some natural language feedback feels very artificial, not natural. i.e. the statement "does it have an group-based assignment" only appears to work if you type it exactly like that. If you type "does it have group assignment", it fails to understand the meaning. Formatting of text feedback needs some form of markup, it tends to appear as walls of text and can be hard to find the required information.*

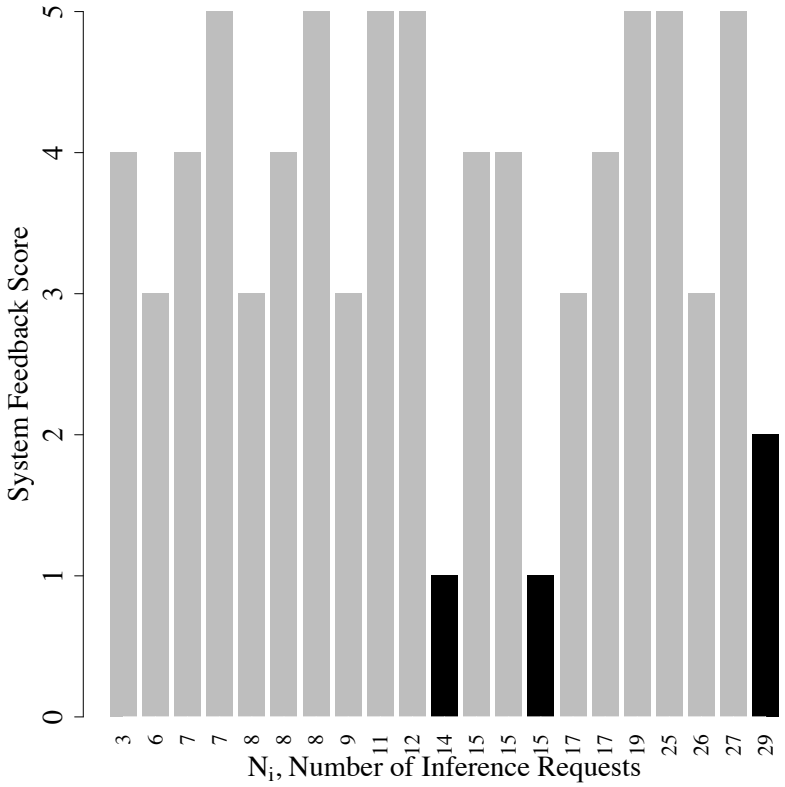


Figure 4.33: System feedback scores by number of inference requests

4.2.10.2 Rule Satisfaction Feedback

Approximately one quarter (26.6%) of individual question responses received feedback ($|N_i| = 127$, out of $|N_{total}| = 478$), and on average, for sessions where feedback was recorded, 45.3% of the session's questions had feedback. This feedback rate may have been improved if the user interface withheld further inference responses until the current response has feedback, but this punitive-style measure could unduly influence the negative feedback response rate.

The total rule feedback score counts attributed to all individually satisfied rules are summarised in Figure 4.34, with the original data presented in Table A.13, Appendix A, and an alternative

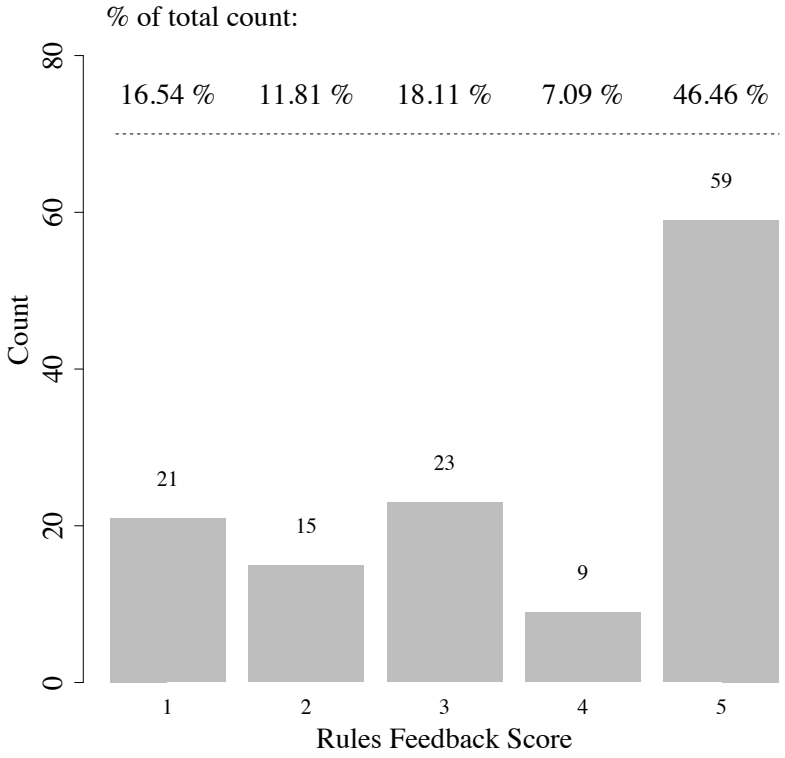


Figure 4.34: Rule feedback score counts

breakdown of rule feedback by user *session* (ordered by the user's number of inference requests, and only for sessions where feedback was received) is listed in Table A.14 in Appendix A.

This result is again very promising, nearly half (46.46%) of the individual response feedback is indicative of score 5, *My answer is exactly what you are seeking*, and if score 4 feedback is included, *My answer is correct but some information is missing*, positive satisfaction rises to 53.55% of response feedback. Dissatisfaction is relatively low – 28.35% of feedback scores (as a result of combining feedback scores 1 and 2 (*My answer is totally wrong* and *My answer is partially wrong*)), and nearly a fifth (18.11%) of feedback was at the ambivalence level (3).

Examining the data in Table A.13, the following observations are made:

- Unsurprisingly the default rule (Rule 0) rates poorly. This is indicative of not receiving a useful response, although 3 sessions interestingly rated this rule as rank 5. Rule 0 achieved the lowest feedback score, but the highest number of feedback responses (38).

- The next highest-feedback count is that for Rule 2 (*unit code*) with 17 feedback responses and its highest frequency (11 out of 17) feedback was score 5.
- Rule 17 (*teaching pattern*) achieved the next highest number of feedback responses (15), with highest frequency (9 out of 15) score of 5. It however also achieved the next highest count of dissatisfied responses (4 of 15) at score 2 perhaps due to misinterpretations of unsupported timetabling questions.
- The fourth highest feedback response is for Rule 1 (*hello*) which received 12 feedback responses and highest frequency of its responses at score 5 (7 out of 9). This is the first non-default rule needing to be satisfied before a conversation can proceed with any further depth.
- The single ambivalent score of 3 for Rules 14 and 23 (*unit weighting* and *software* respectively) are from two separate sessions, the first where the user had a feedback response rate of 29% (5 out of their 17 questions, ID = 37 in Table A.14); the second from a session (ID = 26) with a feedback response rate of 78% (7 of 9 questions).

4.2.10.3 Category-aligned Satisfaction Feedback

Table 4.11: Category-based feedback count totals

Category	$\mathbf{N}_f$	**% Total**
Cat1	75	59.06
Cat2	14	11.02
Cat3	19	14.96
Cat4	19	14.96
Total	**127**	**100**

It is interesting to analyse the breakdown of feedback scores by system response category i.e. associating feedback responses with the validity of the inference request and the system response. Table 4.11 contains the counts of feedback received (N_f) but now attributed to the category of inference request and system response (as defined in Table 4.6).

The counts in Table 4.11 are further investigated by determining the feedback scores themselves for each category, which yields Figures 4.35 to 4.38 (source data can be found in Tables A.15 to A.18 in Appendix A).

Examining Figure 4.35 to Figure 4.38 leads to the following observations:

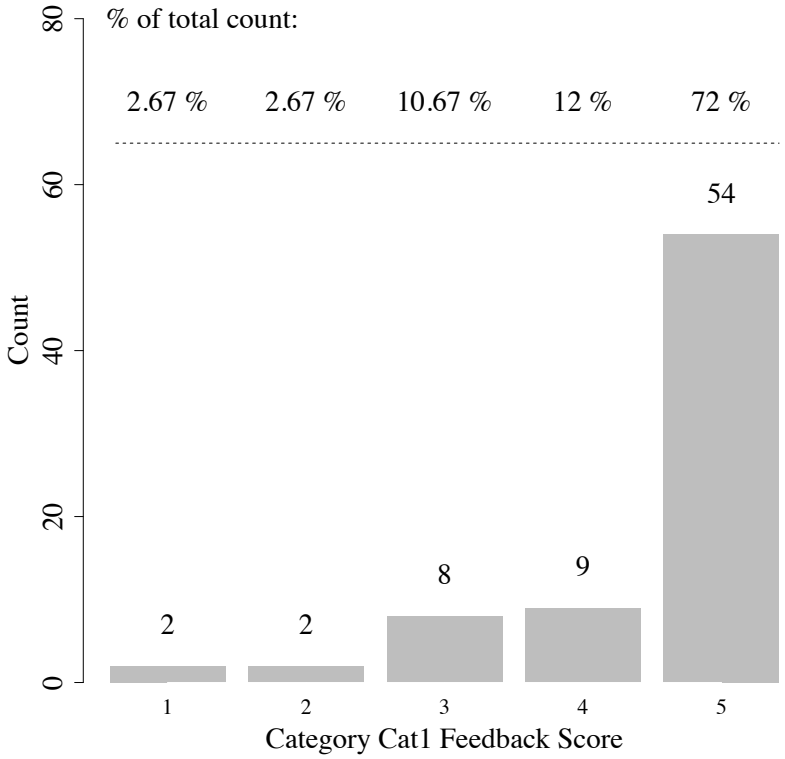


Figure 4.35: Cat1 feedback score counts

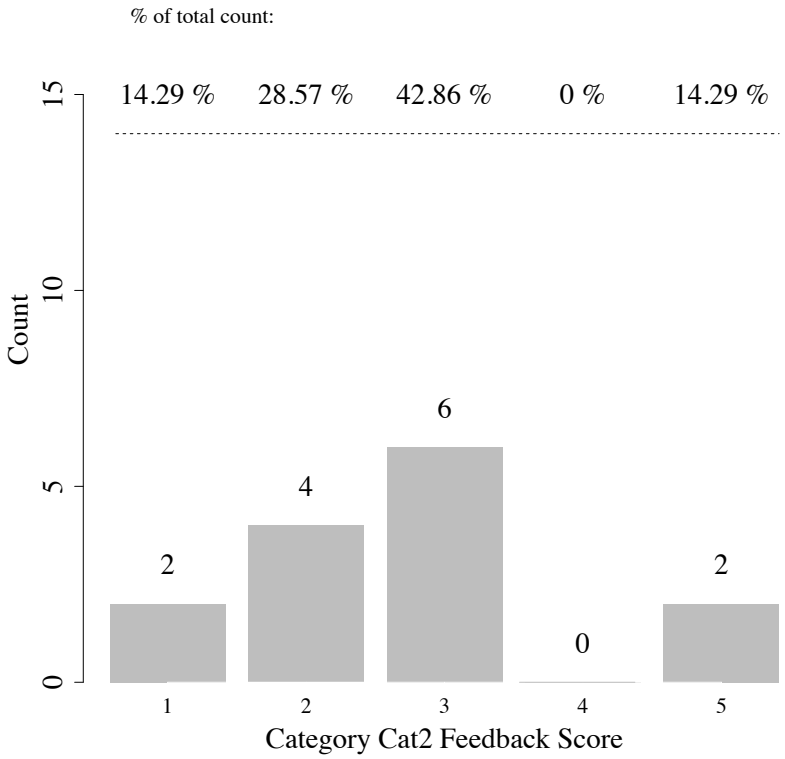


Figure 4.36: Cat2 feedback score counts

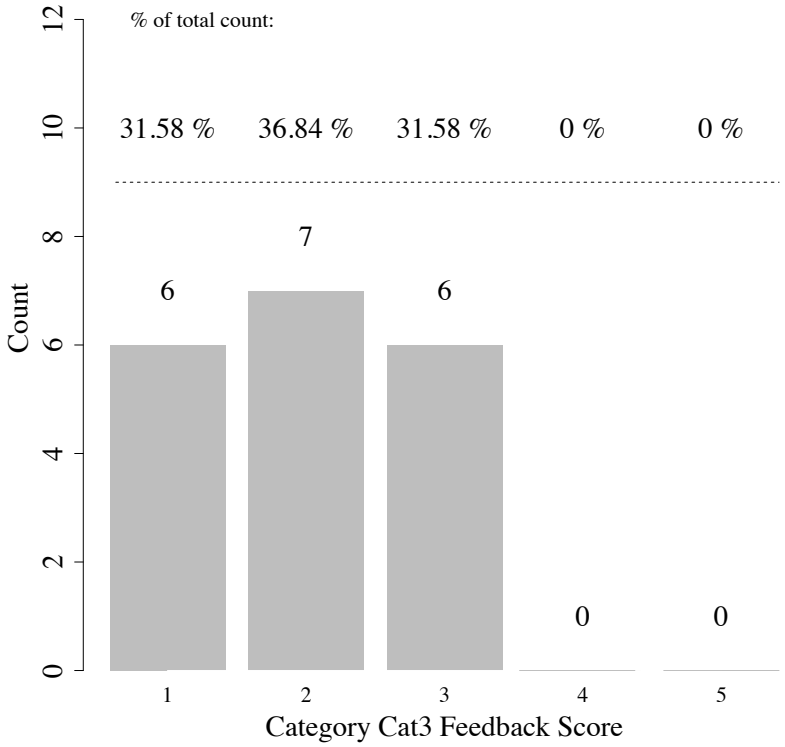


Figure 4.37: Cat3 feedback score counts

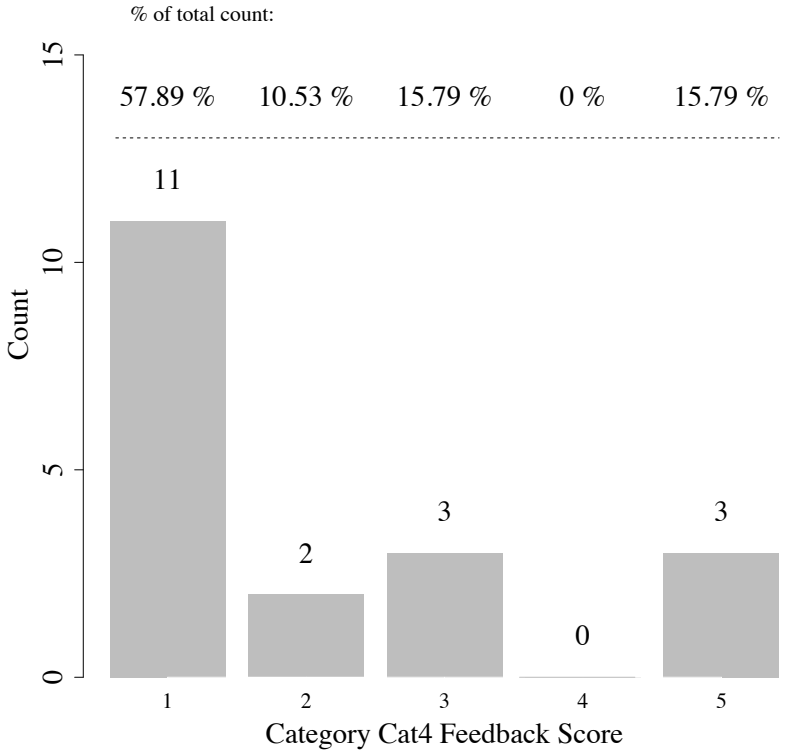


Figure 4.38: Cat4 feedback score counts

- **Category 1, Figure 4.35** – very high satisfaction, 72% (54/75) responses at level 5. This should be the best feedback grouping as here responses are restricted to those associated with only valid question and system responses.

- **Category 2, Figure 4.36** – moderate ambivalence, 42.86% (6/14) responses at level 3, but an equal amount of combined responses at level 1 and 2 (2/7 and 4/7). Here the system responded with a (non-default rule) misinterpretation of the user's question. Interestingly, the low response rate for this category (11.02%) is also perhaps reflected in its highest score count, ambivalence (3).

- **Category 3, Figure 4.37** – moderate to high dissatisfaction, 31.58% (6/19) and 36.84% (7/19) of feedback responses for this category were at score level 1 and 2 respectively. This category also received a higher count of responses, 14.96% of the total. It is unsurprising that 68.42% of responses for this category are dissatisfaction, as from the user's perspective, they have asked a valid question and the system has responded with the default rule, Rule 0. As already discussed, this response is usually due to the user asking a question that is out of scope. From the user's perspective however, they probably assume they have asked a valid question and received an invalid response, hence the low ranking.

- **Category 4, Figure 4.38** – high dissatisfaction, 57.89% (11/19) and 10.53% (2/19) of feedback score 1 and 2 responses. The system is responding appropriately for this category with the default rule, as the user's question is invalid (or non-sensical). Referring to Table A.18, it can be seen the highest contributor to feedback score 1 is the user at 14 inference requests (N=14,ID=5). This user, together with the next highest-contributing user (N=15,ID=34) are also two of the three users identified in Figure 4.33 for rating the system overall below 3 (both rated at score 1 (*I am very dissatisfied*)). Justification for the user's feedback scores would seem to be reflected from this data. Examining their categorised responses (see Table A.19 in Appendix A), the user (N=14,ID=5) had 12 out of 14 (85.71%) category 4 responses. User (N=15,ID=34) had 5 out of 15 category 4 responses (33.33%).

  Counterintuitively, two users ranked three category 4 responses at score 5 – users (N=7,ID=25) and (N=19,ID=38). Perhaps these users realised the system responded appropriately to their invalid questions and thus they ranked them appropriately. User (N=7,ID=25) only ranked 4 responses out of their 7 inference requests (cross-referencing Tables A.14 and A.19, followed by Tables A.15 and A.18), ranking 3 category 1 responses at score 5, and 1 category 4 response also at score 5. User (N=19,ID=38) ranked 2 out of 9 of their category 4 responses at score 5, and 1 other at score 1. They also only ranked 1 other

response, 1 category 1 response at score 5 and had a low feedback response rate, having only ranked the 4 ranks mentioned out of a total of 19 inference requests (21.05%)

The final figure in this section, Figure 4.39 shows the breakdown of feedback scores when considering the *appropriateness* of the system response as defined in Definitions 4.4 and 4.5. The overall positive, negative and ambivalent feedback results, grouped now by *appropriateness* of response, is now considered, noting :

- *Appropriate system responses:* $\sum |N_i| = 94$
  positive feedback: 70.22%, $|N_i| = 66$;
  negative feedback: 18.09%, $|N_i| = 17$;
  ambivalent feedback: 11.70%, $|N_i| = 11$
- *Inappropriate system responses:*$\sum |N_i| = 33$
  positive feedback: 6.06%, $|N_i| = 2$;
  negative feedback: 57.57%, $|N_i| = 19$;
  ambivalent feedback: 36.36%, $|N_i| = 12$

A positive feedback acceptance rate for *appropriate* system responses of 70.22% is very encouraging considering the associated C-MCRDR CA system evaluation knowledge-base size and minimal rule set. The relatively high negative feedback rate of 18.09% possibly stems from the fact that participants may not realise the system is providing an appropriate response (even if it is *I don't understand*), which may lead to some frustration. When considering system misbehaviour, unsurprisingly the combined negative feedback scores are high (57.57%), however it is apparent that users are more reticent to leave feedback for system misbehaviour compared to appropriate behaviour, as this type of response only received about a third of feedback responses ($\sum |N_i| = 33$ compared to $\sum |N_i| = 94$).

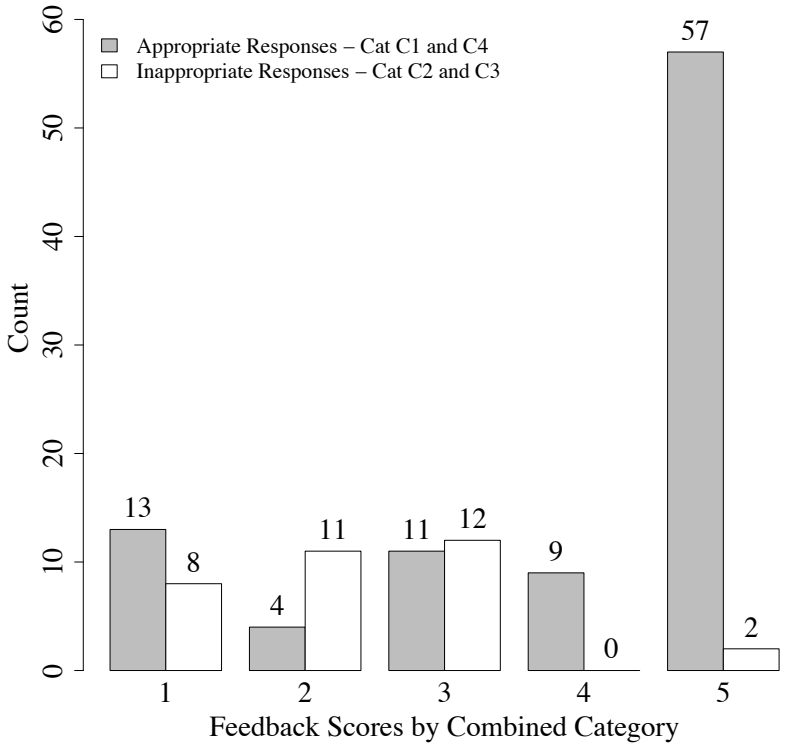


Figure 4.39: Feedback scores by combined category

4.2.10.4 User Acceptance Summary

61.9% of participants who provided overall system performance feedback rated the system positively, and encouragingly, only 14.28% of feedback was in the negative scale. Slightly less than a quarter (23.81%) rated the system at the ambivalence level (*I am neither satisfied nor dissatisfied*). It should be noted these values result from a relatively small sample size ($|N_i| = 21$) and only 3 participants rated the system at the negative levels of 1 or 2, however, as previously stated the sample size is comparable to other similar satisfaction studies (Smith et al., 2014; Shabaz et al., 2015).

When considering user satisfaction for individual system responses (that typically result from single-rule satisfaction) approximately half (46.46%) ($|N_i| = 59$,$|N_{total}| = 127$) of responses are rated at the highest satisfaction level (*My answer is exactly what you are seeking*). Combining with the next positive satisfaction level (*My answer is correct but some information is missing*), yields 53.55% of all feedback responses at the positive level. For dissatisfaction (levels 1 and 2), 28.35% of all feedback responses rate the responses poorly ($|N_i| = 36$).

Manual inspection of each participant inference request and categorising it based on intent and coherency coupled to the feedback received gives more insight into the feedback results, and in particular, causes of negative feedback. When the system is responding appropriately to the participant request, positive feedback rises to 70.22% (of all appropriate system responses that have feedback), $|N_i| = 66$ and negative feedback falls to 18.09%, $|N_i| = 17$. Unsurprisingly when the system responds inappropriately, positive feedback is heavily reduced (6.06% (of all inappropriate system responses that have feedback), $|N_i| = 2$), while negative feedback rises (57.57%, $|N_i| = 19$). The negative feedback results are mainly attributed to out-of-scope questions (which do not produce the requested information) or invalid (non-sensical) questions that then correspond to default-rule responses. The high magnitude of positive feedback responses (best visualised in Figure 4.39) is evaluated as a significant validator of the approach and provides a component of answering research question SRQ2 (defined in Section 1.2.1).

4.3 Intelligent Personal Assistant Evaluation

Section 3.4 in Chapter 3 details the methodology associated with theory, practical setup, and evaluation of two market-leading IPA devices, specifically the Google Home and the Amazon Echo. The results and analysis of the evaluation are detailed in this section and as previously noted they answer the research question:

- SRQ4

  a) *What is the best current market-leading Intelligent Personal Assistant (IPA) to leverage as a speech component of a conversational agent system in terms of ASR performance that allows the default vendor-defined conversational agent to be supplanted?*; and

  b) *How can the IPA's associated ASR errors when coupled to a human-authored rule-based KBS be corrected?*

  (see Section 1.2.1).

This section consists of the following sub-sections:

- Section 4.3.1 Intelligent Personal Assistant Evaluation Environment
- Section 4.3.2 Effects of Source Word Rank
- Section 4.3.3 Effects of Word Length
- Section 4.3.4 Effects of Word Syllable Count
- Section 4.3.5 Effects of Sentence Length
- Section 4.3.6 C-MCRDR WER Improvement

4.3.1 Intelligent Personal Assistant Evaluation Environment

As detailed in Section 3.4.4, each IPA device was evaluated under the same environmental conditions (room, acoustics, speakers and proximity to the device's microphones) between tests. Word lists and sentences were spoken both by the thesis author (an Australian English native speaker) and the Australian *Karen* voice included with Apple macOS X High Sierra 10.13 operating system.

The analysis results looking at factors that affect the recognition performance of the IPA devices tested now follow.

4.3.2 Effects of Source Word Rank

Whether a word's ranking (from the BNC ranking (Kilgarriff, 2006)) has any affect on the word's recognition status is first determined. The effects of rank alone, and then rank coupled with word length are investigated.

4.3.2.1 Effects of Word Rank Only

The recognition results across all letter count categories are aggregated to determine if word rank alone has any significance on the recognition status. Two-sample t-tests were conducted to compare the average word ranking for words recognised and not recognised - please see Tables 4.12 and 4.13 for the Amazon Echo and Google Home tests respectively.

The Amazon Echo result (Table 4.12) shows there was no significant difference in *yes* and *no* word rank means for either the human or Karen speakers, which is indicative that the word rank itself is having no effect on recognition status here.

However, there was some significance in the aggregated ranking data for the Google Home when voiced by a human and computer (see Table 4.13). The two results are further depicted by boxplots in Figures 4.40 and 4.41.

Table 4.12: Amazon Echo t-test for recognition by ranking average

$\mu_{\mathbf{no}}(\sigma_{\mathbf{no}})$	$\mu_{\mathbf{yes}}(\sigma_{\mathbf{yes}})$	**t(df)**	**p**
	Human		
1171.79 (1331.97)	1239.62 (1336.84)	t(593.79)=-0.77	0.44
	Karen		
1225.03 (1387.29)	1214.32 (1277.80)	t(1094)=0.13	0.89

Table 4.13: Google Home t-test for recognition by ranking average

$\mu_{\mathbf{no}}(\sigma_{\mathbf{no}})$	$\mu_{\mathbf{yes}}(\sigma_{\mathbf{yes}})$	**t(df)**	**p**
	Human		
785.28 (1171.37)	1278.37 (1345.56)	(178.13)=-4.42	1.69×10^{-5}
	Karen		
770.11 (1188.84)	1319.05 (1345.84)	t(318.48)=-5.74	2.24×10^{-8}

The effect of word rank on the WER by grouping ranked words into distinct bands of rank value (for rank values increasing by 500 in each band) is now determined. Each rank band contains

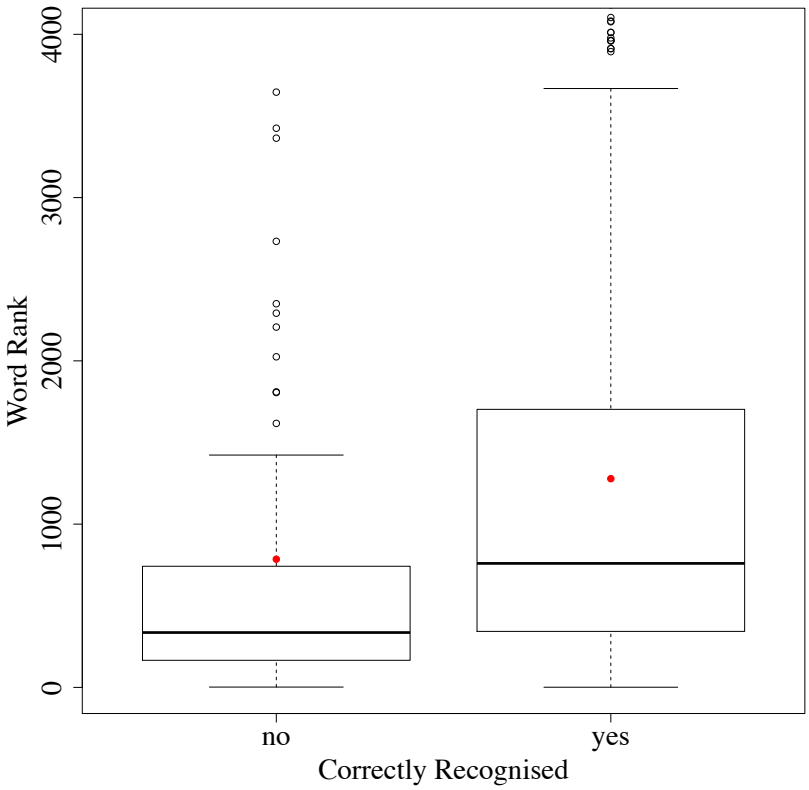


Figure 4.40: Google Home - Human recognition by word rank

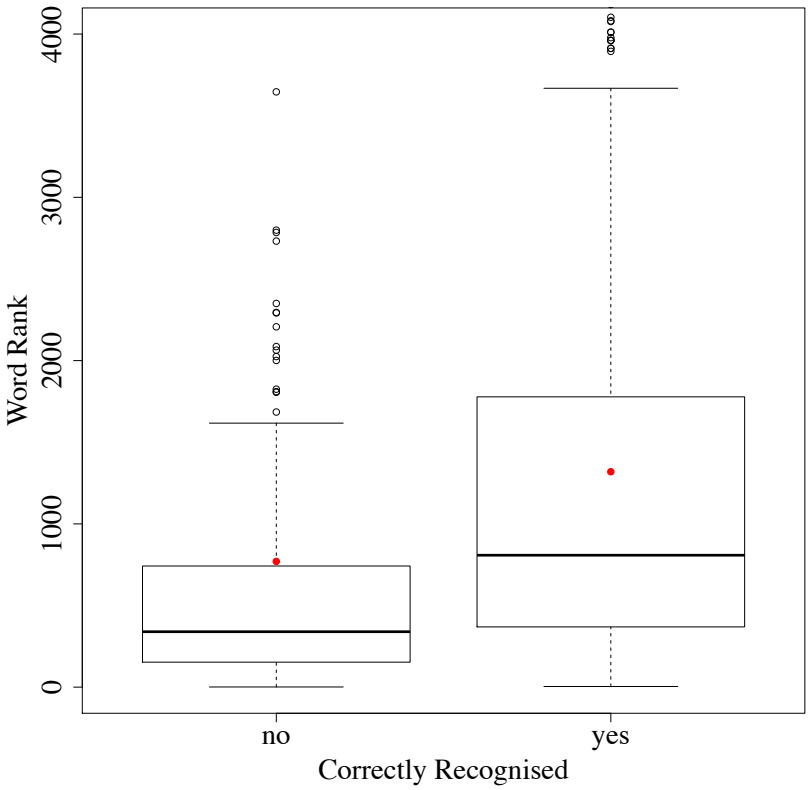


Figure 4.41: Google Home - *Karen* recognition by word rank

words that rank from the previous band's value to the current band – for example, band 3 contains words that rank between 1001 and 1500. The WER calculated for both the human and *Karen* speakers can be seen in Table 4.15. From inspection it is apparent that across the majority of rank bands that:

- WER is worse (unsurprisingly) for *Karen* vocalisations;
- WER is worse for Amazon Echo compared to Google Home

Figures 4.42 and 4.43 show the plots of the WER across all rank bands by device, for a human and for *Karen* respectively. Both Figures show the significant difference in WER between the two devices, and that, for the Google Home, recognition appears to be optimal for words in rank band 6 (word rank between 2501 and 3000) for a human, and rank band 7 (word rank between 3001 and 3500) for *Karen*. It should be noted that rank band 11 onwards only contained 14 words or less, so the WER results for these bands may be anomalous or less significant.

Table 4.14: WER rank band details

Band	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**
Rank	500	1000	1500	2000	2500	3000	3500	4000	4500	5000	5500	6000	6500
Words	424	242	129	83	59	39	32	15	15	29	14	8	7
Google Home with human													
WER	0.18	0.12	0.06	0.04	0.07	0.03	0.06	0.07	0.07	0.10	0	0.13	0.14
Google Home with *Karen*													
WER	0.29	0.14	0.11	0.06	0.14	0.08	0	0.07	0.07	0.10	0.07	0.13	0.43
Amazon Echo with human													
WER	0.33	0.25	0.25	0.29	0.34	0.26	0.34	0.20	0.27	0.41	0.21	0.13	0.14
Amazon Echo with *Karen*													
WER	0.54	0.52	0.50	0.37	0.53	0.49	0.47	0.73	0.73	0.55	0.57	0.50	0.57

The overall average WER across all ranking bands is shown in Table 4.15. As this is in essence an aggregating summary of Table 4.14, the same results are apparent - the computer's vocalisations are not recognised as well as a human's, and Google Home performs better as it has a lower average WER compared to the Amazon Echo.

Table 4.15: Average WER for all rank bands

	Google Home		***Amazon Echo***	
Source	$\boldsymbol{\mu}(\times 10^{-2})$	$\boldsymbol{\sigma}(\times 10^{-2})$	$\boldsymbol{\mu}(\times 10^{-2})$	$\boldsymbol{\sigma}(\times 10^{-2})$
Human	8.08	5.04	26.25	8.16
Karen	12.88	11.31	54.46	9.81

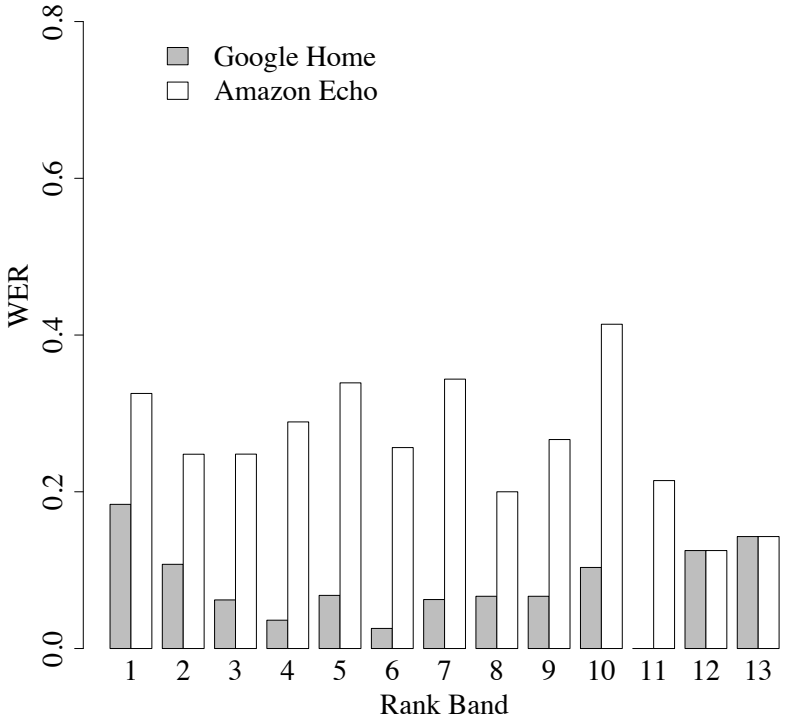


Figure 4.42: WER by rank band for human

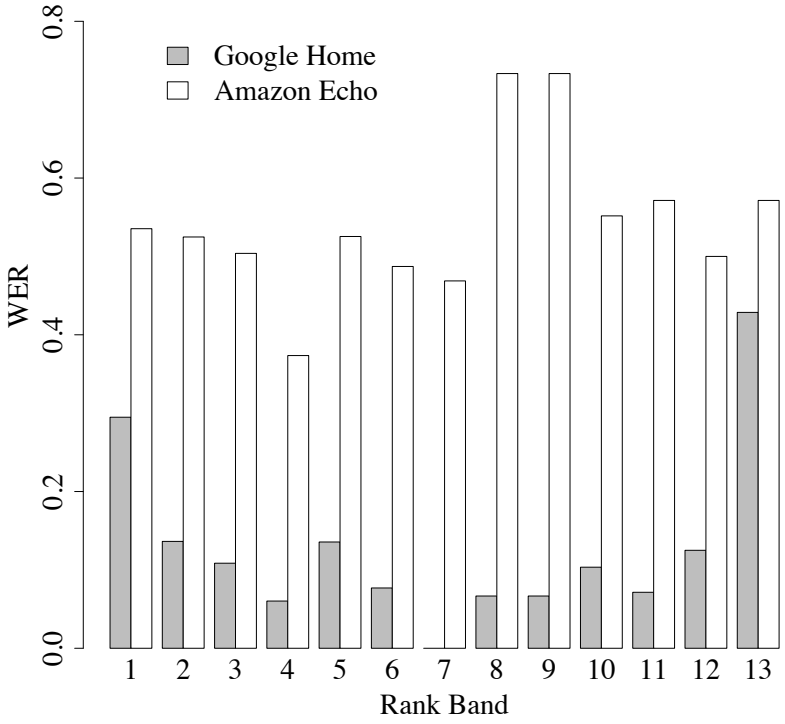


Figure 4.43: WER by rank band for *Karen*

When considering the count of words included in the dataset grouped by their rank in the banded approach detailed above, the distribution frequency can be visualised by a histogram (Figure 4.44). Interestingly, but not perhaps surprisingly, this also follows a power law that is *similar* to Zipf's law (Zipf, 1949) – this was also shown in Section 4.2.4, where rules satisfied during evaluation of the C-MCRDR CA system followed a similar distribution. Figure 4.45 shows the log-log relationship used to determine the linear model which yields the formula (with $R^2 = 0.94$, $p < 5.09 \times 10^{-8}$):

$$W_c = 15.14 \times 10^6 \times \frac{1}{W_r^{\alpha}} \qquad \alpha = 1.62 \tag{4.6}$$

The fit of the linear model found in Equation 4.6 to the histogram from Figure 4.44 is detailed in Figure 4.46.

4.3.2.1.1 Effects of Word Rank Only Summary

The results in this section do strongly suggest the word frequency ranking has an effect on the recognition status for the Google Home (but not the Amazon Echo), but counter-intuitively, the words with low numerical rank (which is indicative of high frequency) have a *negative* effect on recognition. The initial hypothetical assumption that the ASR models for the Google Home are probably trained with higher-frequency words does not seem to hold. The counter argument is perhaps there is a large variability in the ranking values given the high magnitude standard deviations shown in Table 4.13. Another observation is the most frequent, low numerically ranked words chosen in the dataset tend to have a shorter word length, compared to the less frequent, longer word length and high numerically ranked words. Longer word length (and not rank) seems to have a positive effect on recognition, which will be seen in the next section. When grouping words by their rank into bands and determining the WER, the Google Home out-performs the Amazon Echo, for example, the average WER by a human for the Google Home is 8.08×10^{-2}, compared to 26.25×10^{-2} for the Amazon Echo.

4.3.2.2 Effects of Word Rank Considering Word Length

The second analysis examines if the word length and rank has any effect on the IPA recognition status. Tables A.20 and A.21 in Appendix A.4 show the two-sample unpaired t-tests for words of length 2 to 13, voiced by a human for the Google Home and Amazon Echo respectively. In each case, the p-values (set to threshold at 0.05) indicate the alternative hypothesis, namely the *true difference in means is not equal to 0* is not supported, except for the 4 and 10 letter cases for the Google Home. Boxplots of these cases are shown in Figures 4.47 and 4.48. Both plots and their statistical significance seem to support the hypothesis that lower word rank (which

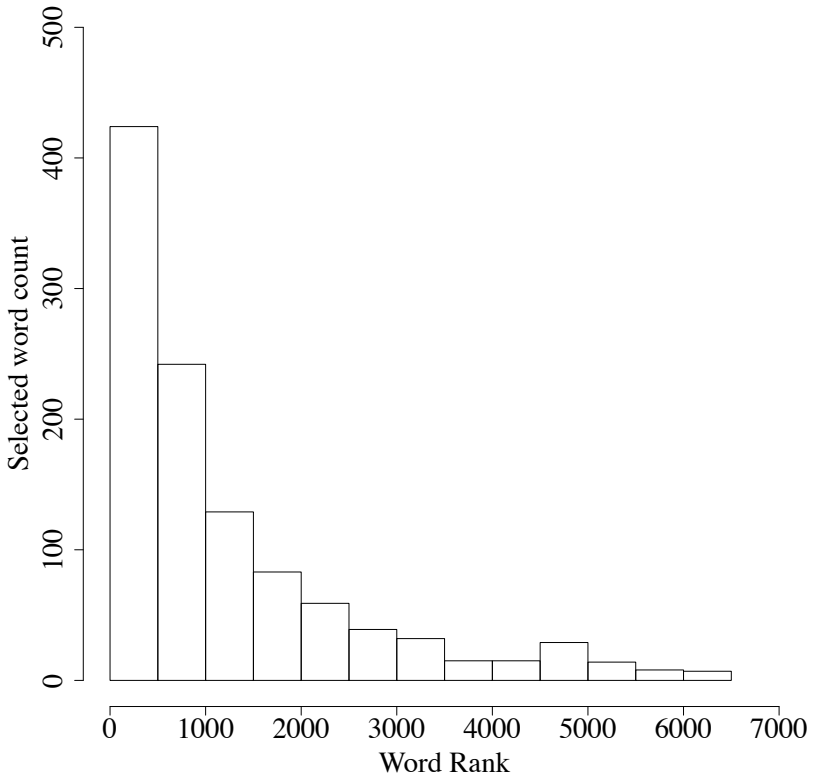


Figure 4.44: Dataset word frequency histogram by rank

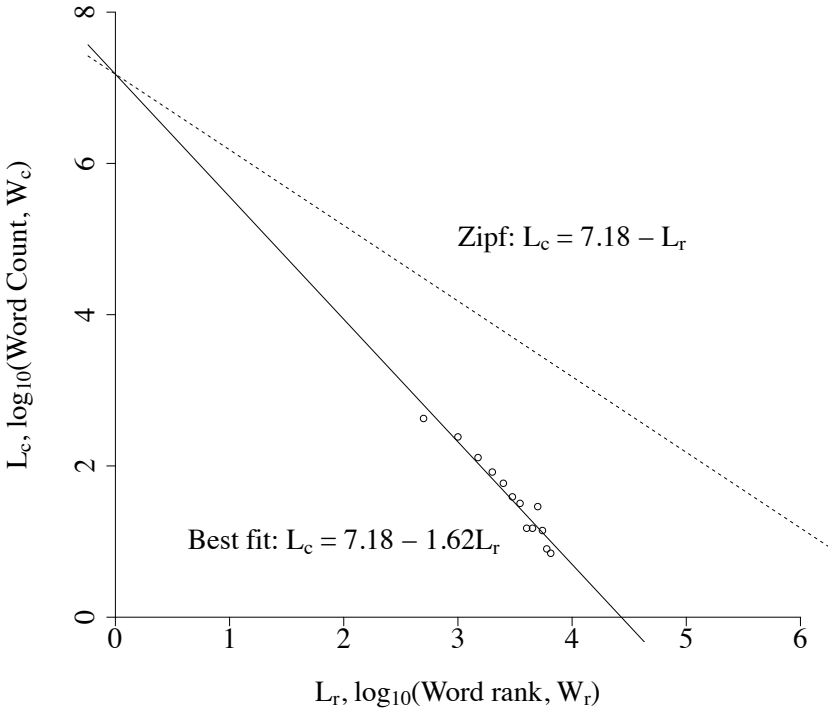


Figure 4.45: Log word count by log rank

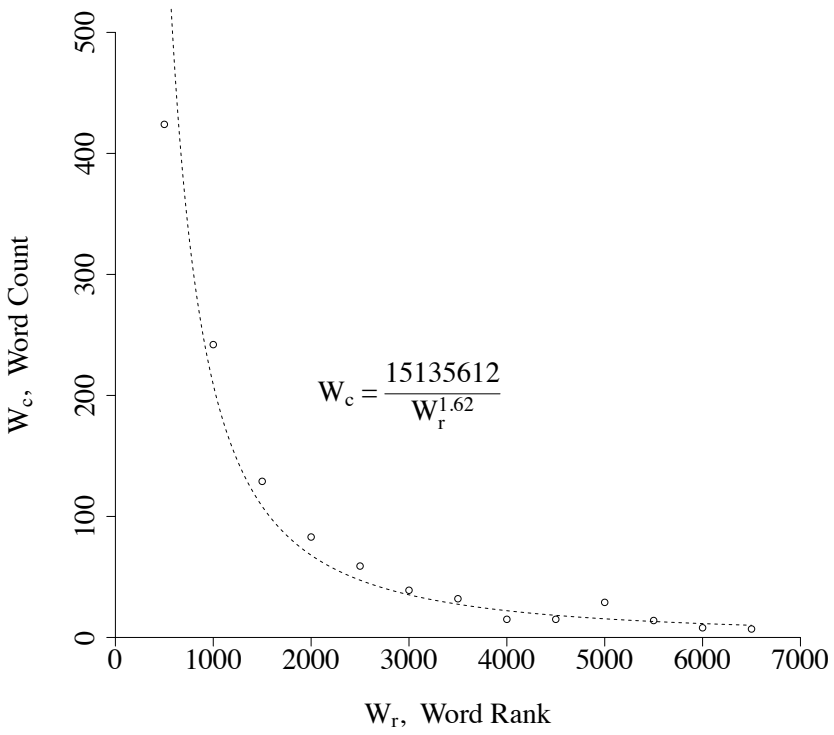


Figure 4.46: Selected word count by word rank (Zipf approximation)

is higher frequency of occurrence) are more likely to be recognised – but only for the Google Home and only for these letter counts and in particular it is slightly more convincing in the 10 letter case (Figure 4.48). A variation in speaker (*Karen* instead of human) has approximately the same behaviour, namely the 4 and 13 letter cases for the Google Home (see Figures 4.49 and 4.50).

The Amazon Echo recognition performance did not seem to be affected by word rank when grouped by letter count and vocalised by a human (Appendix A.4, Table A.21), however there was a significant difference in the *no* and *yes* means for word rank for the 10 letter category when spoken by *Karen*. This is depicted by the boxplot in Figure 4.51. The p-value for this case (2.56×10^{-2}) is not as convincing nor significant as the comparative cases with the Google Home in the 10 and 13 letter cases in Appendix A.4, Tables A.20 and A.22 (3.70×10^{-3} and 4.91×10^{-3} respectively).

In both IPA cases even though there appears to be more convincing significant t-test results for the 10- and 13-letter categories for the Google Home vocalised by a human and *Karen*

respectively, these do appear to be anomalous results. 10-letter words that are more prone to error include singular to plural recognition (2%) and homophones (3%). In the 13-letter category, there is also a 2% homophone error rate, and 2% hyphenated to non-hyphenated error rate. These results appear to be consistent with the other word length data.

The fact that the *yes* mean rank is not significantly different to the *no* mean rank across most letter categories for both IPA devices seems to indicate by this grouping that in general, the rank (grouped by letter count) has no significant affect on the recognition status.

4.3.2.2.1 Effects of Word Rank Considering Word Length Summary

These results suggest in general that there is no significant effect of the BNC word frequency ranking when grouped by word lengths on the recognition status, based on the fact that the *yes* mean ranks are not significantly different to the *no* mean ranks across most letter length categories.

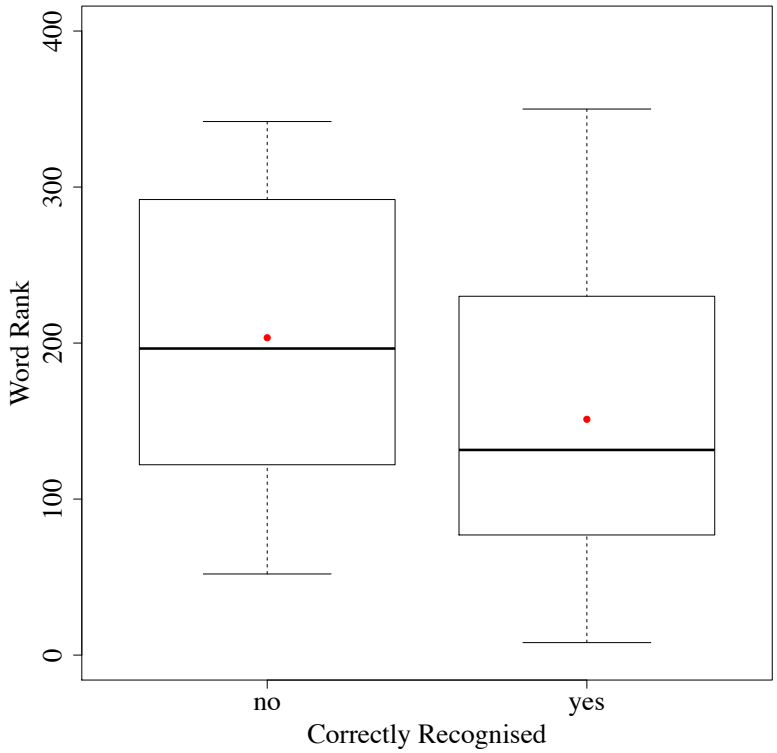


Figure 4.47: Google Home word rank with 4 letters by human

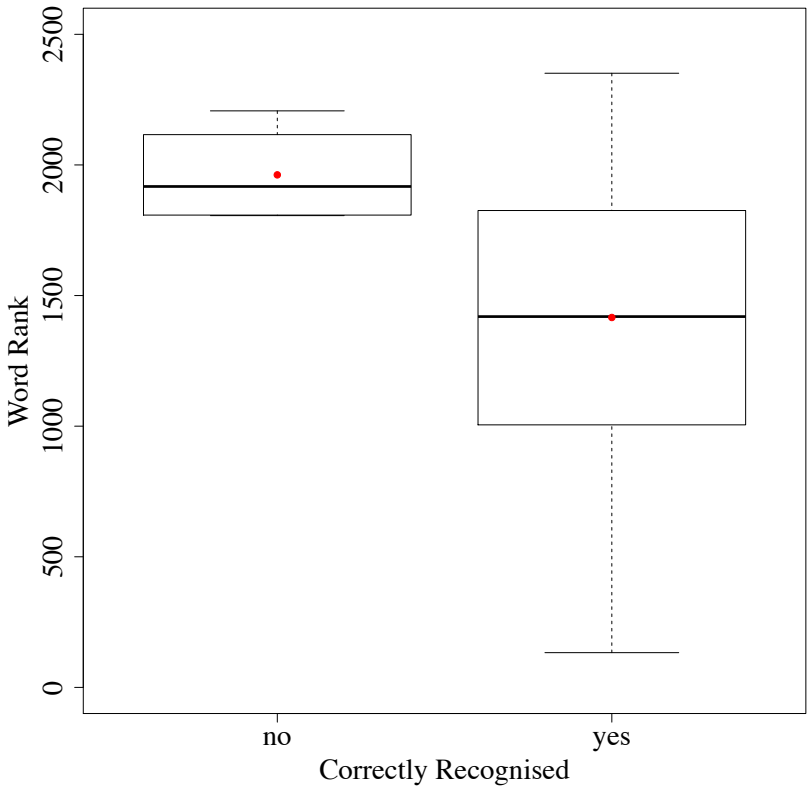


Figure 4.48: Google Home word rank with 10 letters by human

4.3.3 Effects of Word Length

The WER for both devices is now calculated, but categorised now by letter count instead of word rank. There is again a distinct recognition performance difference between the devices which can be seen in Figures 4.52 and 4.53. In both cases, the Google Home consistently has significantly lower WER (8.20×10^{-2}) compared to the Amazon Echo (27.16×10^{-2}) for the human speaker across all letter counts, and unsurprisingly, the WER for both devices suffers when the source vocalisations are by computer, *Karen*. This is not unexpected as *Karen* does not pronounce all words correctly. The mean values associated with Figures 4.52 and 4.53 can be seen in Table 4.17.

Examination of Figure 4.52 for the Amazon Echo in particular seems to indicate some linearity between WER and the word length between 2 and 7 letters. Subsequent linear regression shows this linearity is statistically significant (see Equation 4.7). This relationship is plotted in Figure 4.54 (only the Amazon Echo data is taken from Figure 4.52). Although visual inspection

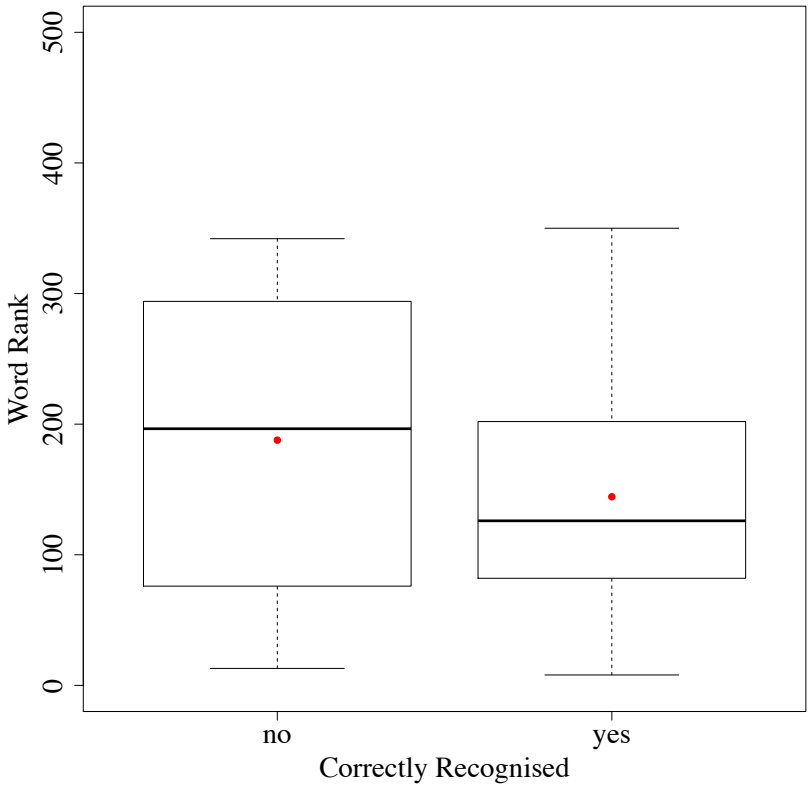


Figure 4.49: Google Home word rank with 4 letters by *Karen*

of Figure 4.53 might suggest a linear relationship between WER and letter count for *Karen* on the Google Home between 4 and 7 or 8 letters, no significance was found.

$$WER_{AmazonEcho-linear} = -0.068N + 0.619, \quad N \in [2,7], p = 9.02 \times 10^{-5}, R^2 = 0.98 \tag{4.7}$$

4.3.3.1 Homophone Recognition in Word Length Effects

Section 4.3.1 indicated homophone responses from both IPA devices were accepted as correct ASR transcriptions, and subsequently it is of interest to ascertain the magnitude of responses that contributed to the WER that were homophones. The percentages of recognised words with a human as the source that are actually homophones are shown in Figure 4.55, with the magnitudes (i.e. number of words) included in Figure 4.56. These results show that if homophones were included as contributors to word errors, then for example for two-letter words

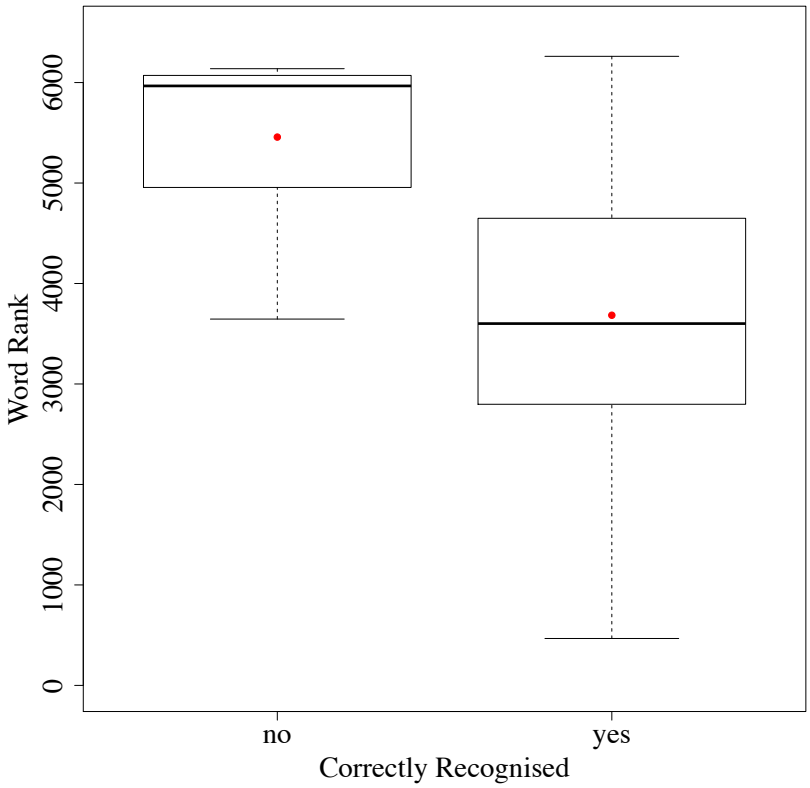


Figure 4.50: Google Home word rank with 13 letters by *Karen*

when vocalised by a human, the WER by letter count data shown in Figure 4.52 would increase from 0.12 to 0.28 for the Google Home, and from 0.48 to 0.64 for the Amazon Echo. For short-letter words, especially 2 or 3 letters, the homophone contribution to the WER (if counted) is quite significant. The mean percentages of homophones accepted across all word lengths for both IPA devices are detailed in Table 4.16. These values show approximately between 4 and 5 percent of words for both devices contribute to homophone recognition (irrespective of word length), which subsequently would require mitigating correction by a global or regional replacement policy when homophones are rejected as erroneous ASR transcription.

Table 4.16: Mean homophone recognition percentages

	Google Home		***Amazon Echo***	
Source	$\boldsymbol{\mu}$(%)	$\boldsymbol{\sigma}$(%)	$\boldsymbol{\mu}$(%)	$\boldsymbol{\sigma}$(%)
Human	4.44	4.58	4.80	4.06

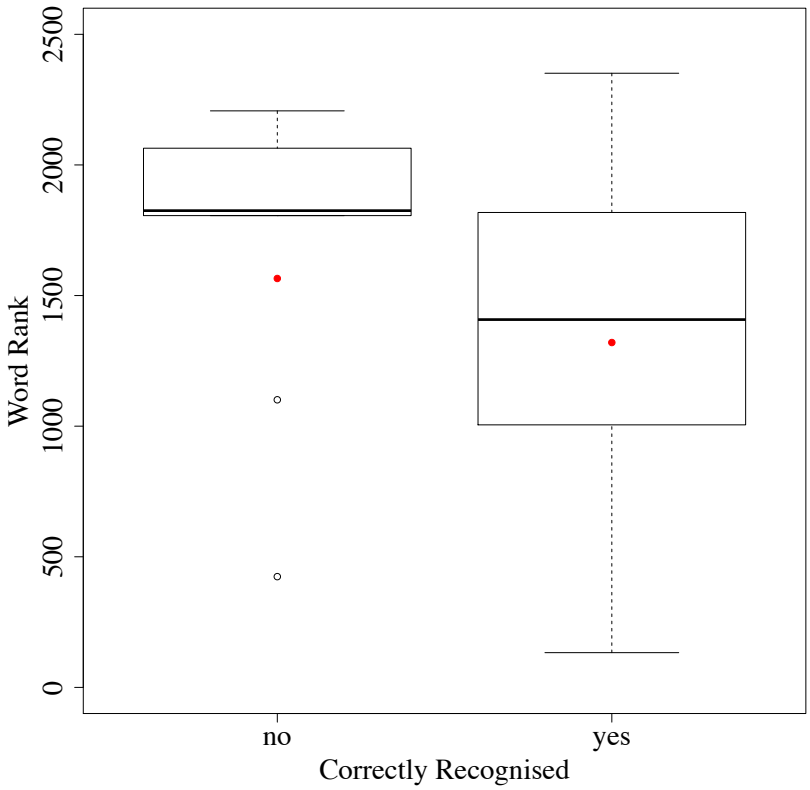


Figure 4.51: Amazon Echo word rank with 10 letters by *Karen*

4.3.3.2 Effects of Word Length Summary

Apart from the unusual linearity effect on WER by letter count for the Amazon Echo with a human speaker, there does not seem to be any significant linear relationship between WER and letter count (some polynomial models can fit the data well, but they are over-fitted). The Google Home continues to show higher performance and its mean WER of 8.2×10^{-2} is a 69.8% improvement over the Amazon Echo WER of 27.16×10^{-2} for a human speaker, and it has a 69.2% improvement in WER based on the computer speaker, *Karen*.

Figure 4.52 suggests single words with word lengths of seven or eight letters are optimal for the Amazon Echo when the speaker is human, whereas the Google Home shows a general improvement of WER with increasing letter count. This appears to be intuitive as increasing letter count generally corresponds to an increase in syllable count and thus word duration, which may be captured more effectively by the devices's microphones.

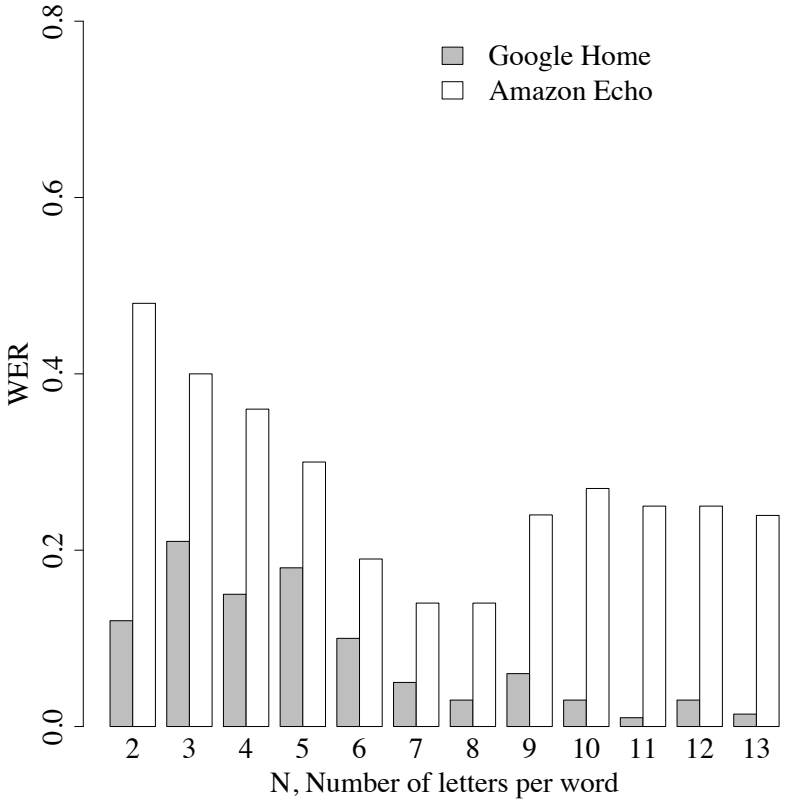


Figure 4.52: Human - WER by letter count

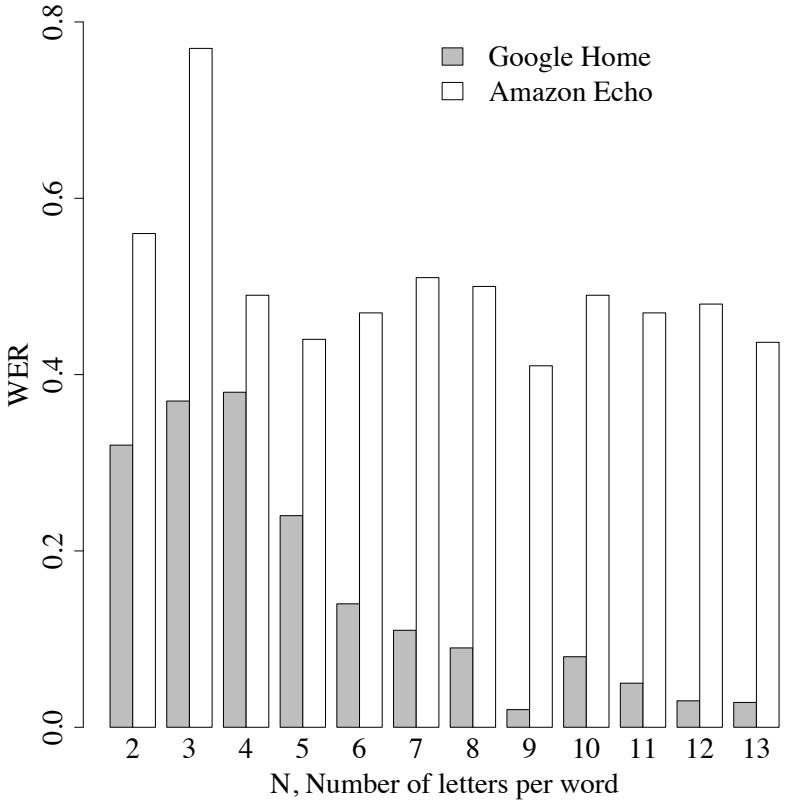


Figure 4.53: *Karen* - WER by letter count

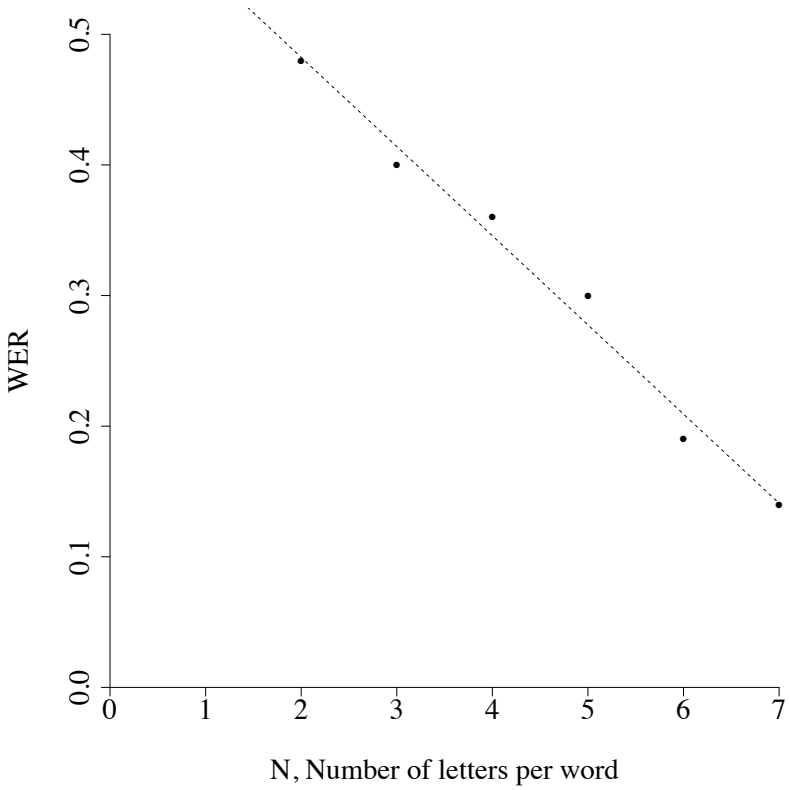


Figure 4.54: $WER_{AmazonEcho-linear}$ model plot

Table 4.17: Mean WER by letter count

	Google Home		***Amazon Echo***	
Source	$\boldsymbol{\mu}(\times 10^{-2})$	$\boldsymbol{\sigma}(\times 10^{-2})$	$\boldsymbol{\mu}(\times 10^{-2})$	$\boldsymbol{\sigma}(\times 10^{-2})$
Human	8.20	6.86	27.16	10.13
Karen	15.48	13.63	50.22	9.28

4.3.4 Effects of Word Syllable Count

Words chosen from the BNC data corpora were manually labelled with their associated syllable count (see Table 4.18) and then the recognition WER for each device and utterance-speaker were plotted against syllable count (Figure 4.57). Note that the WER value for each syllable count plotted here is itself an average of the WER values for every spoken word with the same syllable count.

In Figure 4.57 the same general effects for increasing syllable count are similar to those seen when increasing letter count – a clear separation of performance by Google Home over Amazon

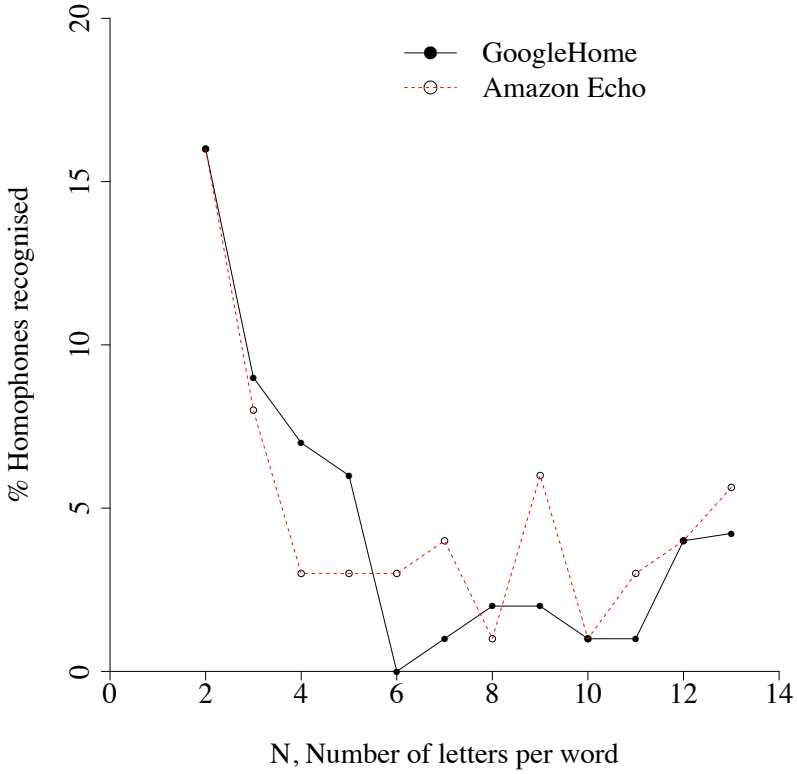


Figure 4.55: Homophone recognition percentage

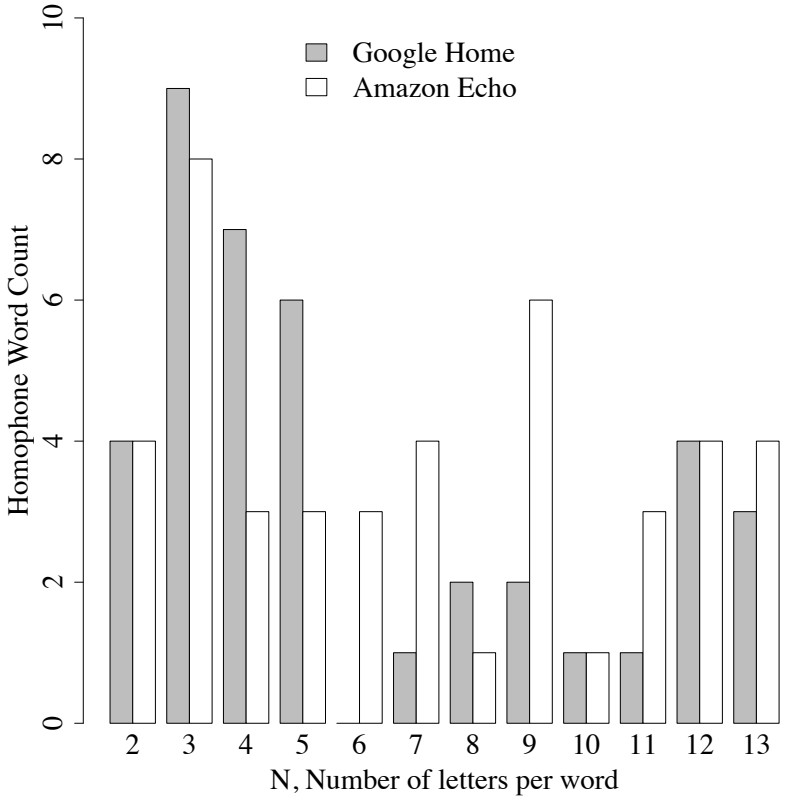


Figure 4.56: Homophone counts by word length

Table 4.18: Syllable frequency for BNC words chosen

	Number of Syllables					
	1	**2**	**3**	**4**	**5**	**6**
Frequency	292	256	251	189	97	11
% of words	26.6	23.4	22.9	17.2	8.9	1.0

Echo, and worse WER performance when the words are uttered by computer, *Karen* (the mean rates can be found in Table 4.19). It should also be noted though that there is not an equal distribution of syllable frequency across all words tested, so the mean WER values shown in Table 4.19 are less robust due to the unequal aggregation of syllable count.

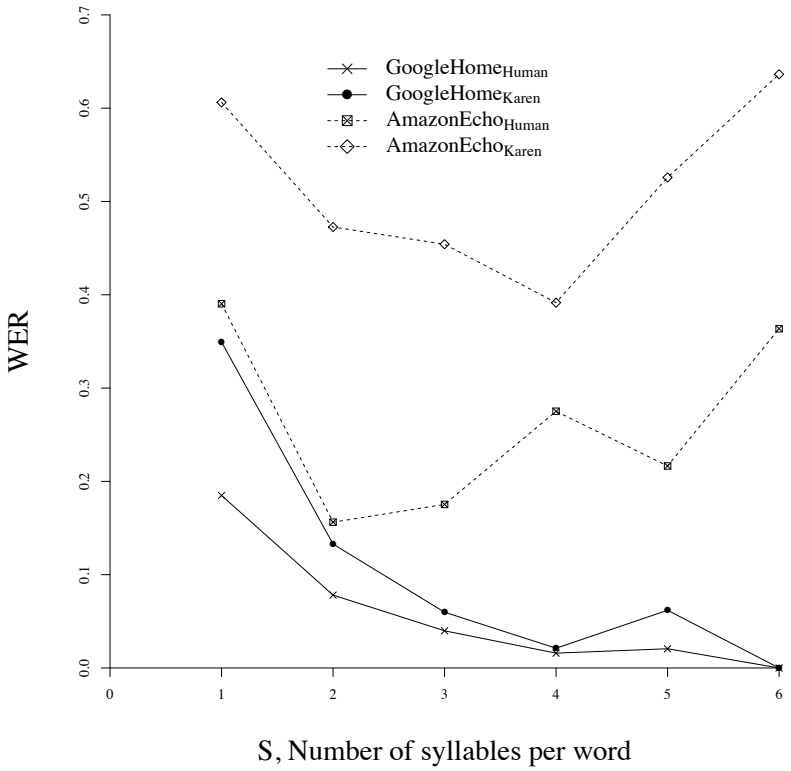


Figure 4.57: WER by syllable count

The Google Home voiced by human WER data is fitted significantly to a polynomial of degree 3 (Equation 4.8) which agrees well with the observed trend of decreasing WER by increased syllable count. The Amazon Echo however, interestingly suffers worse WER with increasing syllable count after 2 syllables for the human speaker and 4 syllables for *Karen*. In contrast to

increasing letter count, here the increase in syllables leads to misclassification by the Amazon Echo.

$$WER_{GoogleHomeHuman} = 0.369 - 0.233S + 5.22 \times 10^{-2}S^2 - 7.59 \times 10^{-4}S^3 \quad (4.8)$$
$$p = 4.78 \times 10^{-3}, R^2 = 0.992$$

It is interesting to note that the mean WER for the Google Home when measured by syllable count is an improvement when compared to the measurement by letter count (5.67×10^{-2} versus 8.20×10^{-2} for human, 10.42×10^{-2} versus 15.48×10^{-2} for *Karen*), with similar standard deviations between the letter and syllable measurements. This may be explained by the fact that the WER for syllables is calculated on larger sets for each data point (for example, there were 292 words with one syllable, 256 words for 2 syllables etcetera – see Table 4.18). In contrast, each letter distribution WER data point is calculated from 100 words (except for the 2 and 13 letter categories) – Table 4.20 indicates the distribution of syllable count across the letter count categories in the dataset.

Table 4.19: Mean WER by syllable count

	Google Home		***Amazon Echo***	
Source	$\boldsymbol{\mu}(\times10^{-2})$	$\boldsymbol{\sigma}(\times10^{-2})$	$\boldsymbol{\mu}(\times10^{-2})$	$\boldsymbol{\sigma}(\times10^{-2})$
Human	5.67	6.84	26.29	9.77
Karen	10.42	12.84	51.44	9.37

Table 4.20: Syllable count by number of letters

	Number of letters											
Syllables	2	3	4	5	6	7	8	9	10	11	12	13
1	24	97	87	64	17	2	1	0	0	0	0	0
2	1	3	11	35	77	74	36	16	3	0	0	0
3	0	0	2	1	6	22	52	64	55	31	11	7
4	0	0	0	0	0	2	11	20	33	54	54	15
5	0	0	0	0	0	0	0	0	9	14	32	42
6	0	0	0	0	0	0	0	0	0	1	3	7
Word count	25	100	100	100	100	100	100	100	100	100	100	71

4.3.4.1 Effects of Word Syllable Count Summary

As expected, the Google Home WER improves with increasing syllable count, exhibiting a mean WER of 5.67×10^{-2} for the human speaker and 10.42×10^{-2} for *Karen*. The Amazon Echo mean

WER is 4.63 and 4.94 times worse for the human and *Karen* speakers respectively. Figure 4.57 suggests in contrast to the Google Home, increasing syllable count *increases* the WER for the Amazon Echo, with optimal WER occurring at words with 2 syllables for a human speaker, and 4 syllables for *Karen*.

4.3.5 Effects of Sentence Length

Sections 4.3.2, 4.3.2.2, 4.3.2.1 and 4.3.3, when considering the WER for isolated words, have established the Google Home is more tolerant to artificially-voiced utterances compared to Amazon Echo (recall Google Home had a mean improvement of 69.8% and 69.2% in its WER for human and *Karen* vocalisations when compared to the Amazon Echo (see Section 4.3.3.2)). In contrast to the earlier measurements associated with differing word lengths and ranking, the *sentence* WER for a single human speaker (the thesis author) against both IPA devices is evaluated. Sentence lengths varied between four and 10 words per sentence, with the evaluation data consisting of 100 instances of each sentence length, and 700 sentences in total.

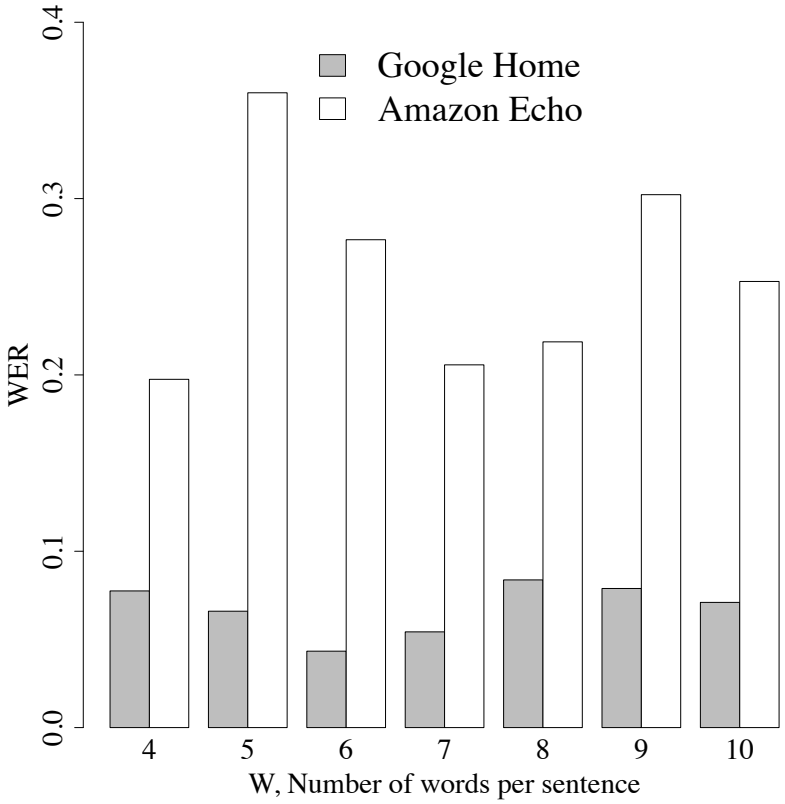


Figure 4.58: WER by sentence length

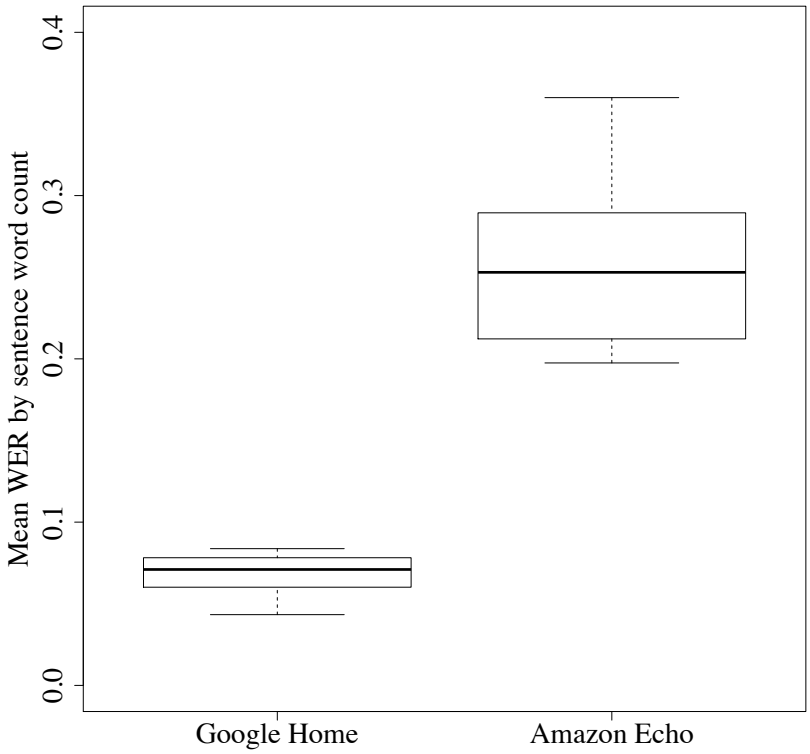


Figure 4.59: Mean sentence WER by device

Referring to Figure 4.58, the WER performance comparison between the two IPA devices is very pronounced – the mean WER for both devices by sentence length is alternatively shown in the boxplot, Figure 4.59, with the overall mean WER across all sentence lengths for the Google home found to be 6.78×10^{-2} ($\sigma = 1.45 \times 10^{-2}$) and the mean sentence WER value for the Amazon Echo was found to be 25.91×10^{-2} ($\sigma = 5.87 \times 10^{-2}$). This plot visually shows the significant difference in mean WER between both devices.

Table 4.21: WER improvement for sentence versus isolated word (human speaker)

Data set	**Google Home** $\boldsymbol{\mu}(\times10^{-2})$	**Amazon Echo** $\boldsymbol{\mu}(\times10^{-2})$
Isolated words (by letter count)	8.20	27.16
Sentences	6.78	25.91
% WER improvement due to sentence context	17.32%	4.60%

There does not appear to be any statistically significant relationship between the WER and sentence word count, however, the lower average WER values compared to *isolated* word eval-

uations show the addition of more context improves the recognition performance. Table 4.21 provides the percentage improvement in WER for both devices when considering the mean isolated word WER and the mean sentence WER, and although small in actual magnitude, Google Home had approximately a ≈17% WER improvement when comparing the isolated word mean to the sentence mean (a difference in WER of 1.42), and the Amazon Echo, although exhibiting approximately the same magnitude of improvement in WER (1.25), had a much lower percentage improvement (4.60%) when contextual information is available in the form of word location within a sentence. Figure 4.58 also indicates the Google Home had an optimal WER for six words per sentence (WER=0.043), and the Amazon Echo's optimal value is four words per sentence (WER=0.2). There is a significant increase in WER for the Amazon Echo for five words onwards (which is its worst WER performance with WER=0.36), and comparing the percentage increase between minima and maxima for both devices independently gives a variation of 93.3% and 82.2% for the Google Home and Amazon Echo respectively, although, it must be noted again that the magnitude of both minima and maxima for both devices differs significantly – for example, comparing the minimum WER value of the Google Home to the minimum WER value for the Amazon Echo gives a percentage increase of ≈355% (0.043 to 0.2), and the corresponding maximum WER comparison increase is ≈330% (0.084 to 0.36).

Finally, Figure 4.60 shows the average magnitudes of the word *edit distances* (number of insertions, deletions and substitutions) by both IPA devices in each sentence length category (which is directly related to the WER) – for example, the Amazon Echo achieved a mean edit distance of 2.72 words when sentence lengths were nine words whereas the Google Home has a mean of 0.71 for the same sentence length. Considering the mean number of words misclassified across all sentence lengths, the Google Home consistently misclassifies with an edit distance of 0.481 words across all sentence lengths, whereas the Amazon Echo has a significantly higher edit distance mean of 1.81 across all sentence lengths – the magnitudes of the mean edit distances can be seen in the boxplot, Figure 4.61. Directly comparing the overall means across all sentence lengths gives a percentage increase of ≈277% relative to the Google Home mean (0.481 compared to the Amazon Echo overall mean WER of 1.812).

4.3.5.1 Effects of Sentence Length Summary

There was no significant statistical relationship between sentence length and WER for either IPA device. The Google Home achieves a mean WER across all sentence lengths of 6.78×10^{-2} with a relatively low standard deviation of 1.45×10^{-2}, compared to the Amazon Echo WER mean of 25.91×10^{-2} with higher variance reflected in a standard deviation of 5.87×10^{-2} – this can be observed in Figures 4.58 and 4.59. Comparing the word *edit distance magnitudes* found

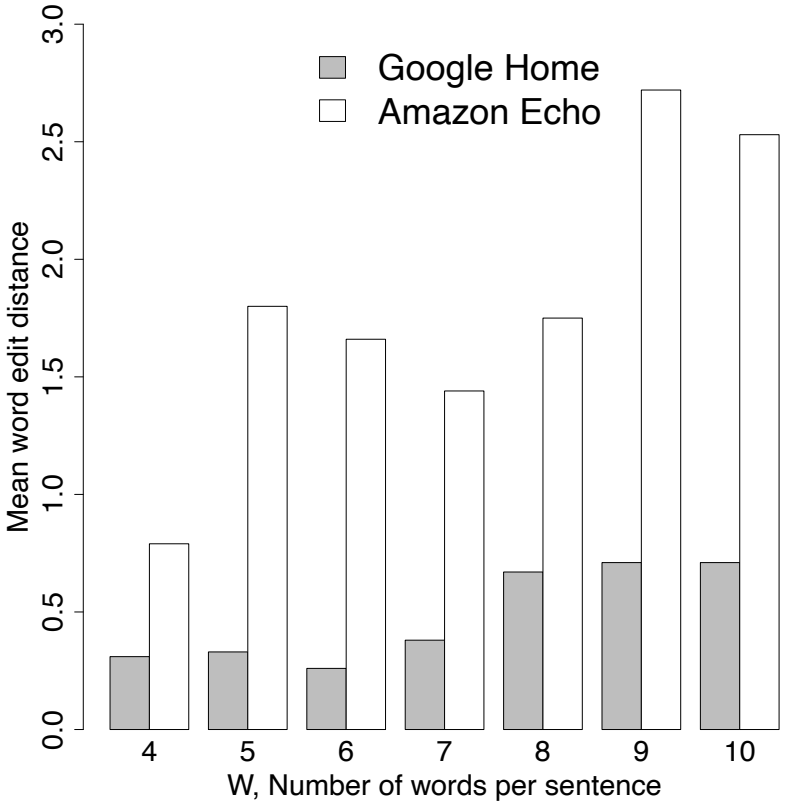


Figure 4.60: Mean edit distance by sentence length

the Google Home had mean edit distance of 0.481 words across all sentence lengths ($\sigma = 0.205$), with the Amazon Echo exhibiting 1.81 words across all sentence lengths ($\sigma = 0.652$).

4.3.6 C-MCRDR WER Improvement

The results from Sections 4.3.2, 4.3.2.2, 4.3.2.1, 4.3.3 and 4.3.5 have a clear indication (at the time of data capture) that the Google Home is the higher performing device based on single-word and sentence WER alone. This device was then chosen to improve its recognition performance *while avoiding homophone recognition of source words* by using a global correction policy to correct utterances before they are presented for inference with the underlying C-MCRDR KBS. Recognition performance for a regional correction policy was then evaluated through introspection of the global correction policy results.

4.3.6.1 Global ASR Correction

Results from this section help to answer research question SRQ4 (see Section 1.2.1).

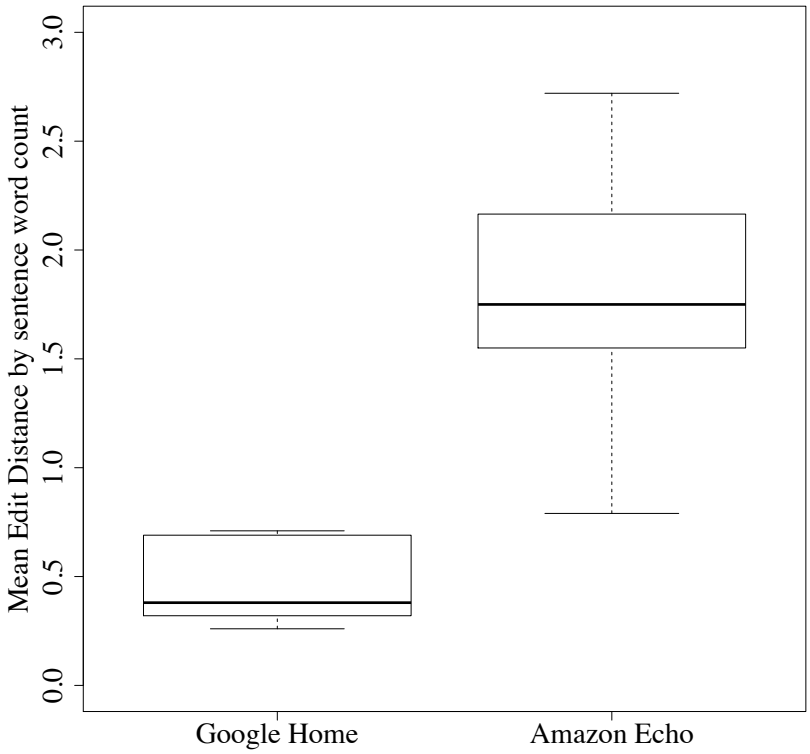


Figure 4.61: Mean sentence edit distance by device

Global corrections are defined (within each word-length category) in successive rounds until the maximum performance recognition rate is achieved. In each table detailing *Correction Round* data, *Correction Round 0* indicates the initial WER, prior to any correction definitions and application. In evaluation each global replacement is simple regular expression defined to match the erroneous ASR error term with the original source word term on a one-to-one basis – please see Sections 3.3.9.1 and 3.4.2.1. In particular, individual e_i are defined in the set RE_G for each word length category.

4.3.6.1.1 Global Corrections Defined

Table 4.22 indicates the number of corrections *defined* (but not necessarily applied) for each round of correction. Due to the slight inconsistencies in word misinterpretations, different words are misinterpreted between rounds, which is a result of a speaker being unable to vocalise an utterance exactly the same each time (O'Shaughnessy, 2008; Errattahi et al., 2018). This means between rounds some defined corrections are not applied as the original errors requiring correction may not have reoccurred. It must be noted the *Total* column data for $N = 2,13$

has been scaled as these two word length datasets are not equal in size compared to the other word length datasets (25 words and 71 words respectively compared to 100 words for the other datasets). In other words, 9 corrections (for 25 words) are scaled to 36 corrections (for theoretical 100 words), and 3 corrections (for 71 words) are scaled to 4 corrections for 100 words. The *Ideal* column in Table 4.22 is defined as the number of global corrections required to reduce the WER to 0 in one round based on the count of ASR substitution or insertion errors that occurred in the initial testing round (*Correction Round 0*); however some word errors in the data set cannot be replaced due to the error term also being present in the dataset. For example, the word *and* is misinterpreted as *end* initially, yet a correction cannot be applied as the word *end* also exists in the dataset – this is a limitation of the Global ASR correction method. This effect means the *Ideal* result does necessarily match the number of corrections defined in correction round 1 for all cases.

Table 4.22: Global word corrections defined

	Correction Round						
N	**1**	**2**	**3**	**4**	**5**	**Total**	**Ideal**
2*	7	1	1			**9→36**	7→28
3	24	11	10	8		**53**	30
4	16	4	8	3	2	**33**	22
5	22	10	8	4	6	**50**	24
6	10	8				**18**	10
7	6	6				**12**	6
8	5	2				**7**	5
9	7	8				**15**	8
10	4	3				**7**	4
11	2					**2**	2
12	5					**5**	7
13*	3					**3→4**	4→6

Table 4.23: Global 4-letter word rules, N=4, Round 1

RE_G	**Term**	**Replacement**
e_1	no	know
e_2	sun	some
e_3	bike	like
e_4	Kia	here
e_5	sing	seem
e_6	h	each
e_7	March	much
e_8	hi	high
e_9	weak	week
e_{10}	meat	meet
e_{11}	lime	line
e_{12}	no	name
e_{13}	our	hour
e_{14}	raped	rate
e_{15}	Les	less
e_{16}	sought	sort

Table 4.23 columns *Term* and *Replacement* indicates the 16 simple regular expression rules defined in *Correction Round* 1 for 4-letter words, ($N = 4$) as an illustrative example. Aligning these correction rules with methodological definitions from Section 3.4.2.1 (specifically Equation 3.38) is shown in the RE_G column. Note that each word length evaluation is performed independently of the others, so the T_s and t_n defined in Section 3.4.2.1, Equation 3.39 is defined as the 100-word utterance set for the given value of N.

Figures 4.62 (actual data) and 4.63 (fitted model), corresponding to $N = 3, 4, 5$ in Tables 4.22 and 4.24 show the actual number of RE_G e_i corrections that were defined for each correction round, noting that between each round the actual misinterpreted words from the source corpora differ. In total, $|RE_{G3}| = 62$, $|RE_{G4}| = 42$ and $|RE_{G5}| = 77$. The fitted logarithmic models shown in Figure 4.63 were only significant for the 3 and 5 letter categories:

$$D_3 = 23.07 - 13.03 \times log_e(C), R^2 = 0.885, p = 1.04 \times 10^{-2}$$
$$D_5 = 20.05 - 10.49 \times log_e(C), R^2 = 0.853, p = 1.61 \times 10^{-2}$$

Figures 4.64 and 4.65 are a visual summary of the *Total* and *Ideal* columns from Table 4.22. Here they indicate the required correction definition totals, but across all word lengths. Below $N = 6$ there is considerable variability or inconsistency in misinterpretation values, meaning a combination of pronunciation differences or phoneme stress produces different results. As detailed previously, the ideal result is defined as the number of global corrections needed to

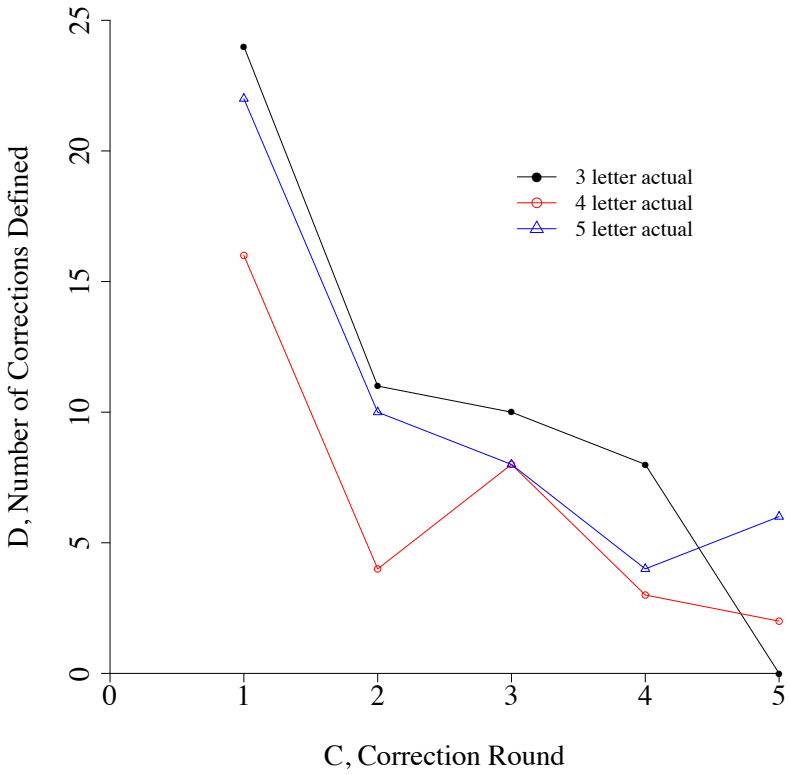


Figure 4.62: RE_G e_i defined for N=3,4,5 actual

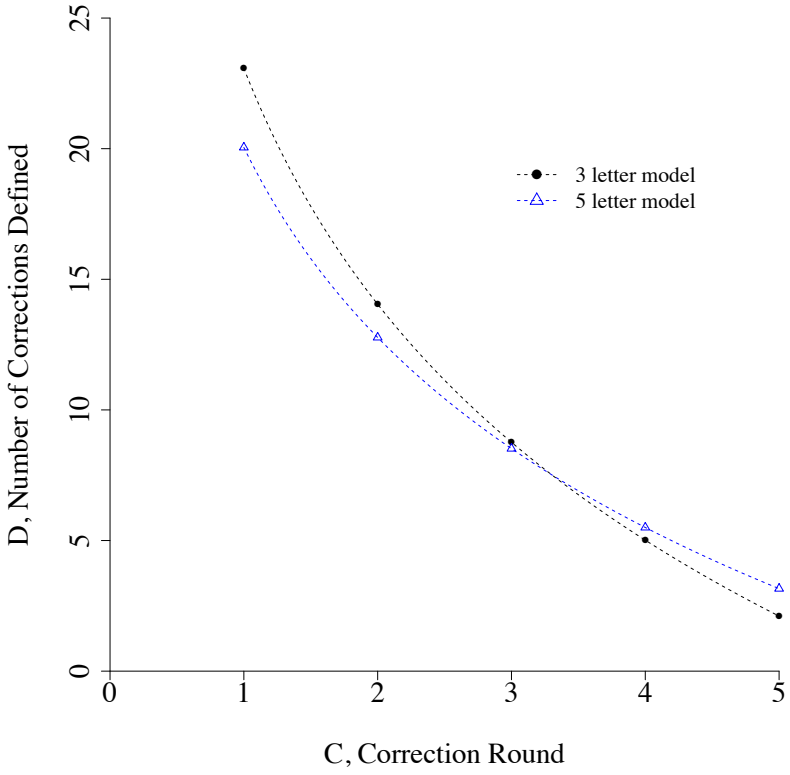


Figure 4.63: RE_G e_i defined for N=3,5 model

reduce the WER to 0 in one round. The set of words defined by the global correction policy should be the union of the sets of words found across each round in order to ameliorate the variability of recognition. This is highlighted by the results shown in Table 4.24 – in all cases (apart from $N = 10$), the actual number of applied corrections in the final respective correction round is less than or equal to the ideal number. This means some words that were previously misinterpreted are now correctly interpreted *without replacement*. Ideally, when using a global or regional correction policy, only small subsets of the word corpora would be scrutinised (for example, keywords associated with pattern matching terms in the C-MCRDR conversation system rule criteria). Figures 4.64 and 4.65 also show the statistically significant non-linear models fitted to the data, although it must be noted the relatively low R^2 value of the model for T, indicating the (observable) high variability of the data, specifically for $N < 6$:

$$T = 69.36 - 26.17 \times log_e(N), R^2 = 0.679, p = 6.04 \times 10^{-4}$$
$$I = 42.48 - 15.86 \times log_e(N), R^2 = 0.809, p = 4.25 \times 10^{-5}$$

These results depicted in Figures 4.64 and 4.65 do seem intuitive, by showing longer length words have less recognition errors and consequently lower WER values (as seen later in Table 4.25 in Section 4.3.6.1.4), and it thus follows that they require less corrections to be defined.

4.3.6.1.2 Global Corrections Applied

Table 4.24 shows the actual number of corrections applied each round, for example, in *Correction Round* 4 for $N = 3$, 20 corrections were applied (lowering the WER to 0.06, which will be seen in the corresponding row in Table 4.25 that follows in Section 4.3.6.1.4). Here, as mentioned above, each correction round is also now potentially correcting words that were previously successfully recognised but due to natural variation in pitch, speed of delivery etcetera have now been misinterpreted by the IPA's associated agent (Google Assistant). Continuing with the example ($N = 3$), 20 corrections were actually required to mitigate ASR substitutions or insertions in round 4, but only another 8 corrections had been defined (as detailed previously in Table 4.22). A visualisation of the data from Table 4.24 can be seen in Figure 4.66 for $N = 3, 4, 5$.

4.3.6.1.3 Global Correction Ratio of Applied to Defined

It is interesting to consider the ratio of global corrections applied each correction round to the number of global corrections cumulatively defined (Figure 4.67). On initial visual inspection it would appear that between 30% to 60% of the cumulatively defined global corrections are being applied each round, and this is confirmed by calculating the mean ratio value for each correction round, which has been plotted as the red data points in the figure (the blue data

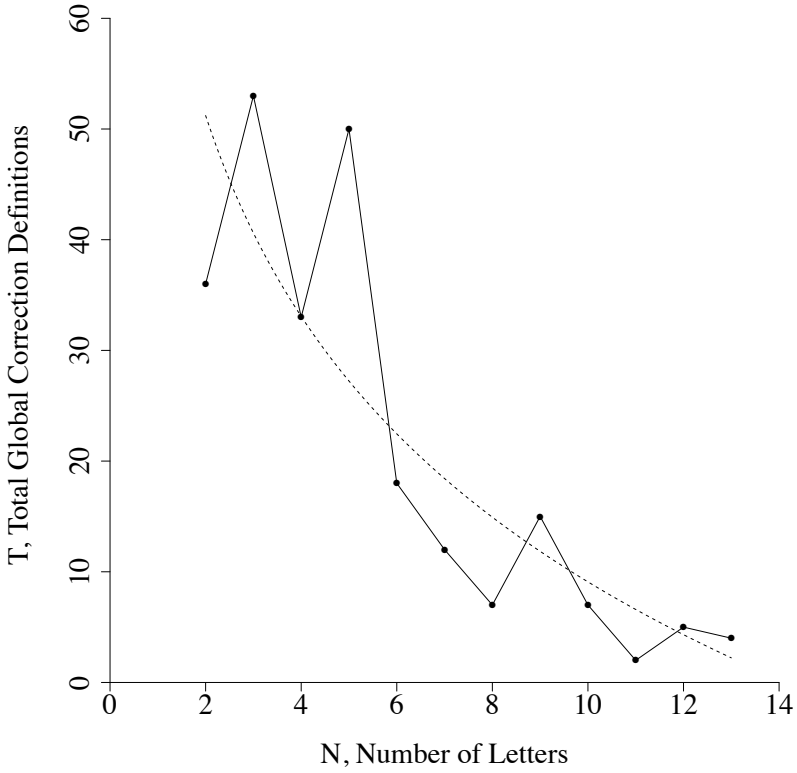


Figure 4.64: Total global corrections defined by letter count($\forall N$)

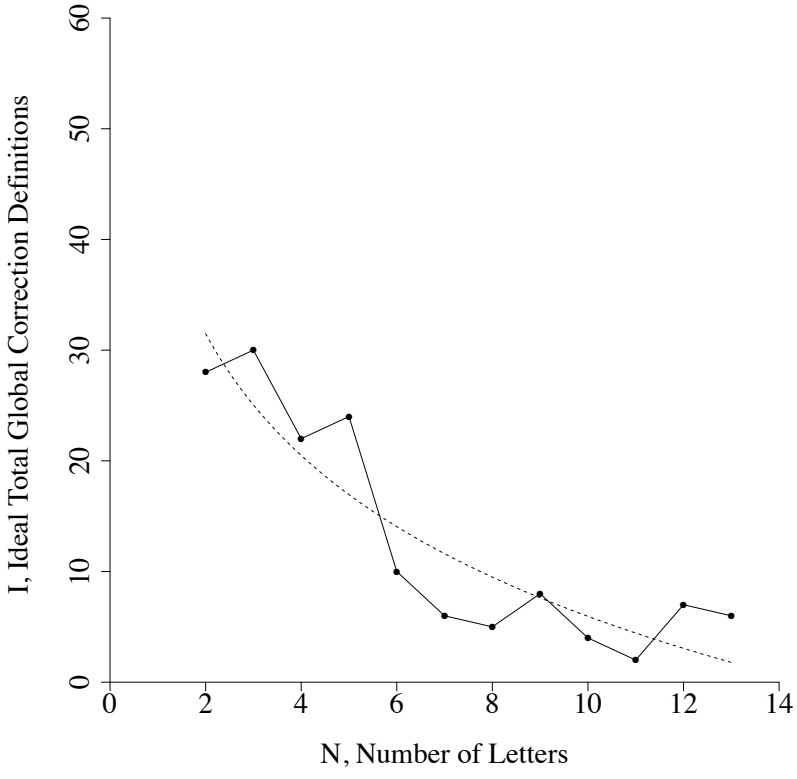


Figure 4.65: Total ideal global corrections defined by letter count

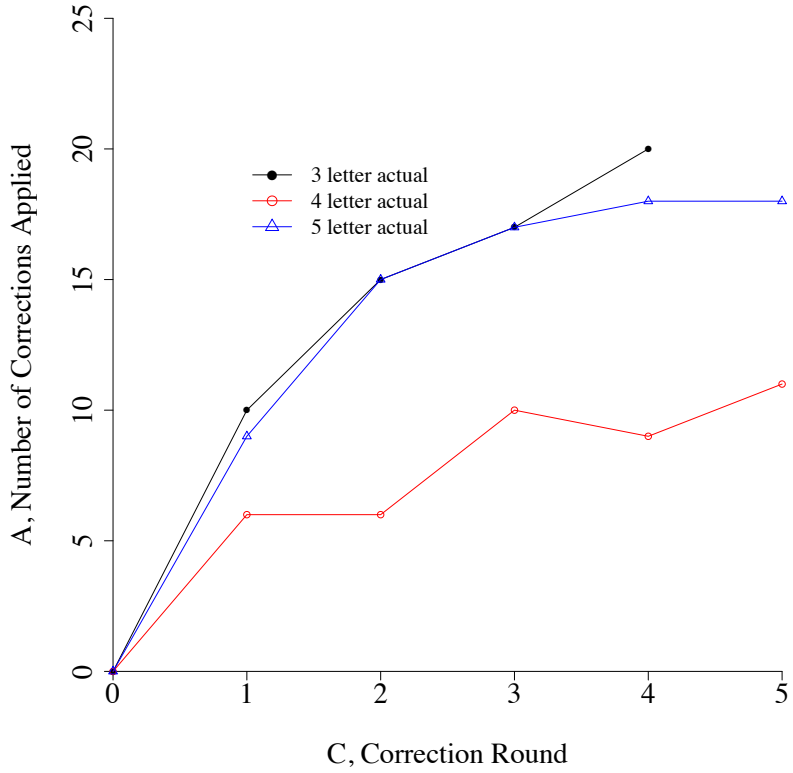


Figure 4.66: Global corrections applied by correction round for $N = 3, 4, 5$

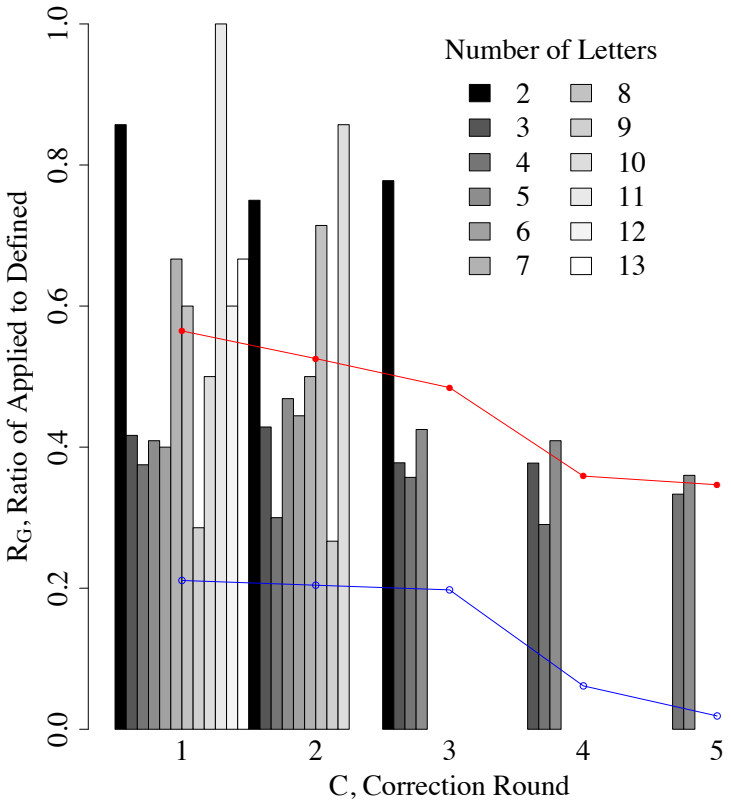


Figure 4.67: Ratio of global applied to global defined by correction round

Table 4.24: Global word corrections applied

	Correction Round				
N	**1**	**2**	**3**	**4**	**5**
2	6	6	7		
3	10	15	17	20	
4	6	6	10	9	11
5	9	15	17	18	18
6	4	8			
7	4	6			
8	3	5			
9	2	4			
10	2	6			
11	2				
12	3				
13	2				

points indicate the standard deviation). The mean ratios also fit a statistically significant linear model which is also intuitive - as more rounds of correction are included, more correcting rules are defined, but the application of those rules is a smaller subset of all definitions:

$$R_G = 0.61 - 5.02 \times 10^{-2} \times C, R^2 = 0.914, p = 7.13 \times 10^{-3} \tag{4.9}$$

Inspection of the mean data values reveals the average ratio for applied corrections to defined corrections is 0.56 (or 56%) for correction round 1, so approximately half the number of global corrections defined for round 1 are actually applied (on average). The greatest actual ratio is associated with $N = 11$ in round 1, where all definitions are applied, but this must be tempered by the fact that the magnitude is small – only 2 corrections were defined and applied. By correction round 5 the mean ratio is 0.35, although only 2 word length categories ($N = 4, 5$) contribute to the mean for this round. The global policy ratio mean (for all correction rounds and word lengths) is defined as the *effort* is $\mu_G = 0.507$ and standard deviation, $\sigma_G = 0.196$.

4.3.6.1.4 Global ASR Correction Performance

The full results across each word-length category (now filtered by only considering a human speaker) can be seen in Table 4.25. Here it can be seen for example, 5 rounds of global correction were needed in the 5 letter length category, to bring the WER to the best possible value, 0.03, although only one round of global correction is required for $N = 11, 12, 13$.

The performance data for three word length categories ($N = 3, 4, 5$) that required multiple rounds of correction from Table 4.22 is visualised in Figures 4.68 (actual values) and 4.69

Table 4.25: WER global improvement by correction round

	Correction Round					
N	**0**	**1**	**2**	**3**	**4**	**5**
2	0.28	0.04	0.04	**0**	-	-
3	0.3	0.15	0.18	0.14	**0.06**	-
4	0.22	0.09	0.11	0.08	0.05	**0.04**
5	0.24	0.14	0.12	0.07	0.11	**0.03**
6	0.1	0.08	**0**	-	-	-
7	0.06	0.07	**0.01**	-	-	-
8	0.05	0.02	**0**	-	-	-
9	0.08	0.09	**0.01**	-	-	-
10	0.04	0.03	**0**	-	-	-
11	0.02	**0**	-	-	-	-
12	0.07	**0.02**	-	-	-	-
13	0.06	**0.01**	-	-	-	-

(fitted model values). Here it can be seen the initial pre-replacement WER values of 0.3, 0.22 and 0.24 for the three to five word lengths respectively are improved with statistically significant fitted logarithmic models after five rounds of mis-recognised utterance global correction.

$$WER_3 = 0.286 - 0.123 \times log_e(1 + C), R^2 = 0.843, p = 6.16 \times 10^{-3}$$
$$WER_4 = 0.199 - 0.091 \times log_e(1 + C), R^2 = 0.838, p = 6.63 \times 10^{-3}$$
$$WER_5 = 0.229 - 0.101 \times log_e(1 + C), R^2 = 0.849, p = 5.57 \times 10^{-3}$$

In each case the minimal WER is achieved but is non-zero due to the use of the global correction policy with words in the source corpora that are corrected as a consequence of other misinterpreted term corrections, for example in the four letter category the following corrections are problematic as the correction terms already exist : *yeah→year, then→than, here→hear* and *turn→term* – this would be avoided with a regional correction policy but this was not implemented in the evaluation system, however, as previously mentioned, the regional correction policy is evaluated in the next section through introspection of the global correction policy data.

Figure 4.70 indicates the total number of corrections applied (totalled across all word length categories) whereas Figure 4.71 provides a visual summary of the global C-MCRDR correction method applied to the source BNC data grouped by letter count for the Google Home with a human speaker. A maximum of five correction rounds were required to minimise the WER across all letter count groups (66.67%, eight out of twelve groups, were minimised after only two correction rounds). The red'|' character in this figure indicates the round where the WER was first minimised for each specific letter count. It can be seen that the WER was initially highest

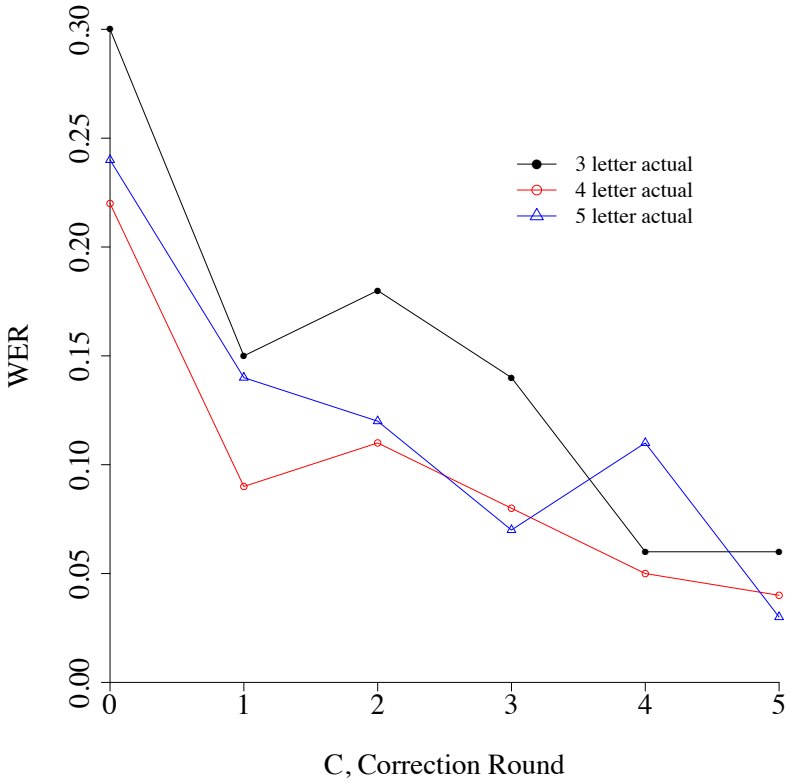


Figure 4.68: WER improvement over 5 rounds for N=3,4,5 actual

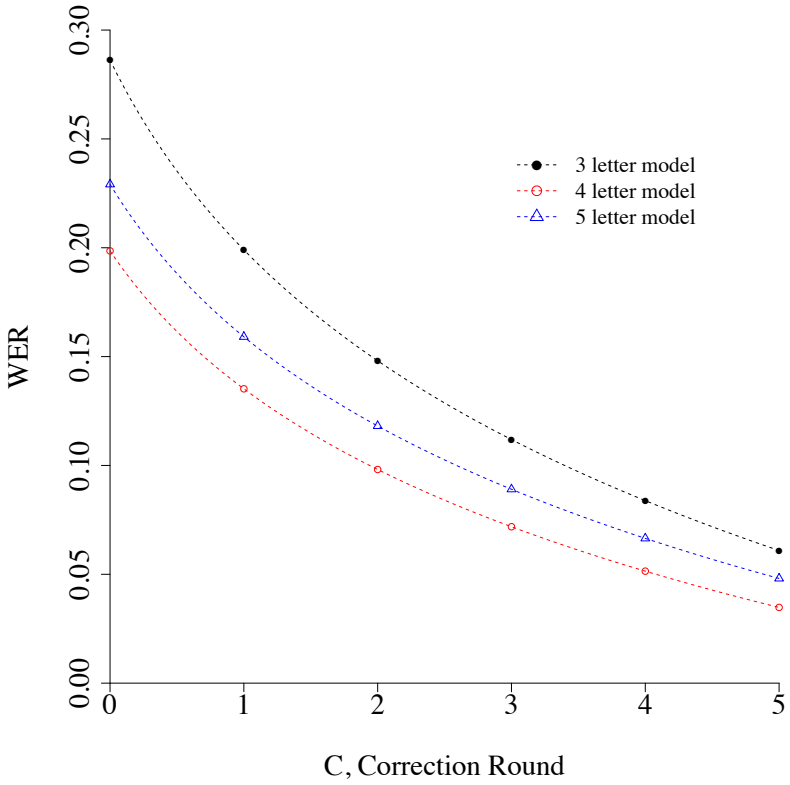


Figure 4.69: WER improvement over 5 rounds for N=3,4,5 model

for the 2-5 letter count words – these contributed the most to the grouped average WER in each round. The WER was not reduced to zero in all letter-length groups as seven of the groups contained words from the source corpora that could not be corrected (the ASR transcription errors produced words that are also present in the corpora, so they cannot be globally replaced). In a production system, in theory the WER can be reduced to zero in all cases where regional context is incorporated. Table 4.26 shows the final, average improvement of the WER for the Google Home after five correction rounds, which shows the global replacement achieves an 87.9% improvement in the WER. It should be noted here the initial WER for *Correction round* 0 is 12.64×10^{-2} (compared to 8.20×10^{-2} from Table 4.21) as now source word homophones are treated as erroneous and thus the WER in increased in comparison.

Table 4.26: Global WER improvement by correction round

	Google Home	
Correction round	$\boldsymbol{\mu}(\times10^{-2})$	$\boldsymbol{\sigma}(\times10^{-2})$
0 (no correction)	12.64	10.24
1	6.20	4.95
2	4.20	6.04
3	2.87	4.46
4	2.28	3.40
5	1.53	1.92
WER Improvement (0 to 5)	87.9%	

4.3.6.2 Regional ASR Correction

Results from this section help to answer research question SRQ4 (see Section 1.2.1).

The regional correction policy methodology (Section 3.4.2.2) was not implemented in the C-MCRDR CA system, nor in the IPA testing environment. However, such a policy can be introspectively evaluated from the global correction policy evaluation data if the assumption is made that the global regular expression terms $e_i \in RE_G$ from Equation 3.38 defined in the previous section, Section 4.3.6.1, are also analogously defined as regional regular expressions, $e_i'' \in RE_r$ (see Equation 3.43). Actual implementation would require the e_i'' rules to be defined in suitable subtrees $T_x \subseteq T_a$ as specified in Equation 3.45.

4.3.6.2.1 Regional Corrections Defined

In an analogous way to Table 4.22 in Section 4.3.6.1.1, the count of regional corrections defined per correction round is displayed in Table 4.27, with the corresponding global value shown in

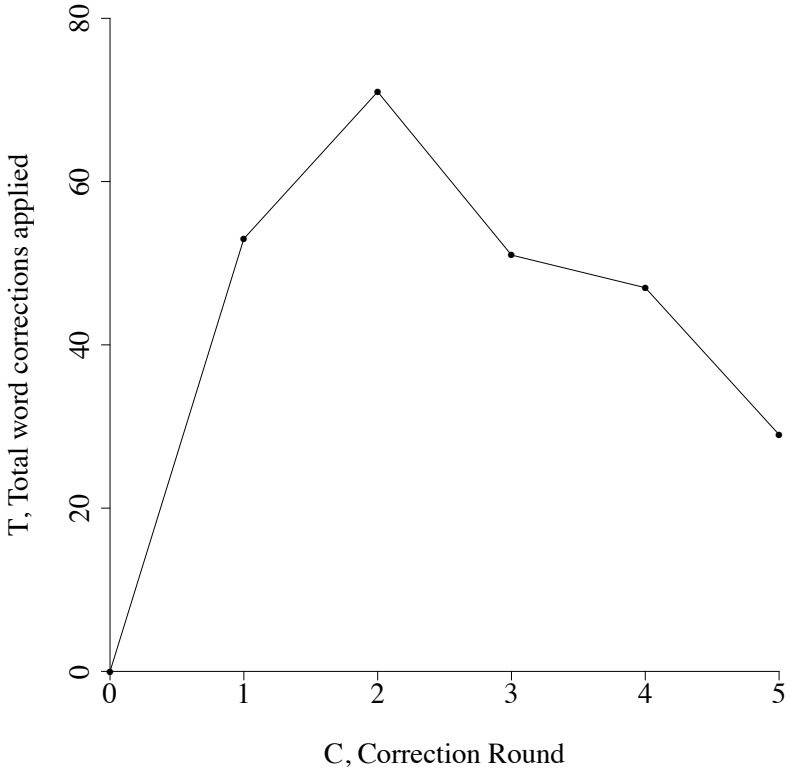


Figure 4.70: Total corrections applied ($\forall N$)

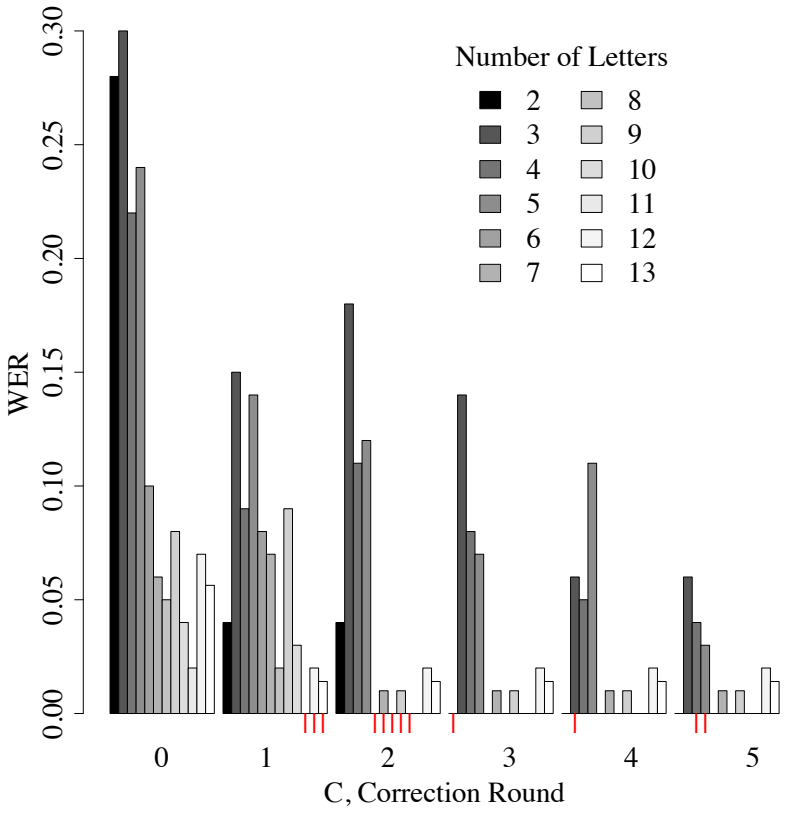


Figure 4.71: WER improvement by letter count across correction rounds

parentheses. Here the count totals for $N = 2, 13$ are again scaled to give an indicative value if the data sets were increased to 100 words.

Table 4.27: Regional word corrections defined

	Correction Round					
N	**1**	**2**	**3**	**4**	**5**	**Total**
2*	7 (7)	1 (1)	1 (1)			**9→36** (**9→36**)
3	29 (24)	13 (11)	13 (10)	8 (8)		**63** (**53**)
4	22 (16)	5 (4)	8 (8)	4 (3)	2 (2)	**41** (**33**)
5	24 (22)	11 (10)	8 (8)	4 (4)	7 (6)	**54** (**50**)
6	10 (10)	8 (8)				**18** (**18**)
7	6 (6)	6 (6)				**12** (**12**)
8	5 (5)	2 (2)				**7** (**7**)
9	8 (7)	8 (8)				**16** (**15**)
10	4 (4)	3 (3)				**7** (**7**)
11	2 (2)					**2** (**2**)
12	7 (5)					**7** (**5**)
13*	4 (3)					**4→6** (**3→4**)

Table 4.28: Regional 4-letter word rules, N=4, Round 1

RE_r	**Term**	**Replacement**
e''_1	no	know
e''_2	sun	some
e''_3	**yeah**	**year**
e''_4	**then**	**than**
e''_5	bike	like
e''_6	Kia	here
e''_7	sing	seem
e''_8	h	each
e''_9	March	much
e''_{10}	hi	high
e''_{11}	weak	week
e''_{12}	**home**	**hold**
e''_{13}	**talk**	**book**
e''_{14}	**here**	**hear**
e''_{15}	meat	meet
e''_{16}	lime	line
e''_{17}	no	name
e''_{18}	our	hour
e''_{19}	raped	rate
e''_{20}	Les	less
e''_{21}	**turn**	**term**
e''_{22}	sought	sort

Table 4.28 indicates the regional replacement rules for $N = 4$ in *Correction Round* 1 in RE_r as an example, with the terms that differ to the global RE_G set shown in bold. In contrast to the corresponding global policy data (Table 4.23), an additional 6 rules are defined in the regional policy.

4.3.6.2.2 Regional Corrections Applied

As the regional policy allows more replacement definition rules to be defined that would not be possible in a global policy, there is subsequently an increase in the rules actually applied during evaluation. Figure 4.72 shows the totals of all definitions applied (for all word length categories) across each correction round, with the corresponding global policy totals also shown for comparison. Figure 4.73 breaks down the regional application totals for the $N = 3, 4, 5$ word length categories (as they required the greatest number of correction rounds to reduce the WER to minimal values), and it includes the global policy applied data (as the dashed lines). It can be seen that the regional policy has had the greatest effect on $N = 3$ and $N = 4$ word length categories as they diverge from the global policy values by the largest magnitudes. The inclusion of extra correction rules (in comparison to the global policy evaluation) as a mitigating policy to cover more transcription errors has an obvious effect on the WER – this is demonstrated next in Section 4.3.6.2.4.

4.3.6.2.3 Regional Correction Ratio of Applied to Defined

With the speculative re-evaluation of the global correction policy data to determine the application of correction rules that constitute a regional correction policy, there is an adjustment to the ratio of rules applied versus rules defined that was demonstrated in Section 4.3.6.1.3 for the global policy. Here, Figure 4.74 displays the ratios by banded data for each word length category against the correction rounds, together with the mean ratio value (in red) and standard deviation from the mean (in blue). The mean ratios for each correction round also fit a statistically significant linear model:

$$R_r = 0.64 - 5.41 \times 10^{-2} \times C, R^2 = 0.946, p = 3.49 \times 10^{-3} \tag{4.10}$$

Comparing the global ratio model (from Equation 4.9, $R_G = 0.61 - 5.02 \times 10^{-2} \times C$) to the regional ratio model R_r above demonstrates the regional model has a steeper gradient (5.41×10^{-2}), a difference compared to the global ratio model of 7.21%. This result is again intuitive – a regional correction policy allows more correction rules to be defined (and applied), but the ratio shows over time the number of rules being applied decreases in comparison to the number of rules defined, and the effect is slightly more pronounced for the regional policy due to the fact

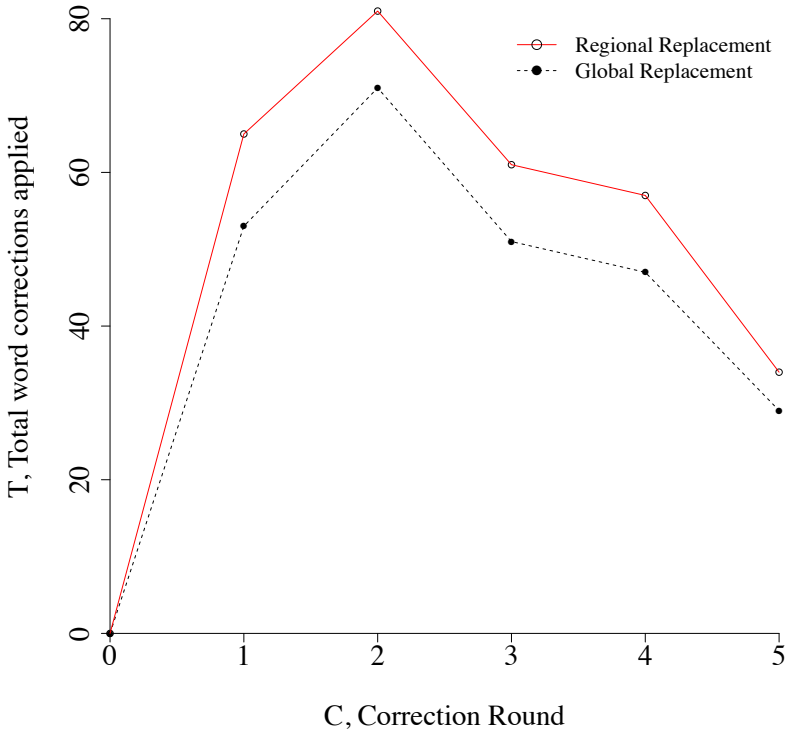


Figure 4.72: Total regional corrections applied ($\forall N$)

that more rules are able to be defined compared to a global policy alone. The regional policy ratio mean (the *effort*, for all correction rounds and word lengths) is $\mu_r = 0.529$ and standard deviation, $\sigma_r = 0.190$. Comparing this to the global policy ratio mean ($\mu_G = 0.507$) it can be seen overall there is a 4.2% difference. In essence, a ratio of 1 indicates all defined rules are being applied, and the smaller the ratio the greater the effort (i.e. more rules are being defined that are not actually being applied on average). The regional policy mean ratio of the number of applied rules against the number of defined rules at 0.529 means slightly less than 50% of the effort, defining regional correction rules, is required to minimise the WER (please see the next section).

4.3.6.2.4 Regional ASR Correction Performance

Table 4.29 details the WER improvement by correction round for each word length category, but the data analysis now assumes a regional replacement policy is applied. In particular, the WER is improved for $N = 3, 4, 5, 9, 12, 13$ as under the global replacement policy, these datasets

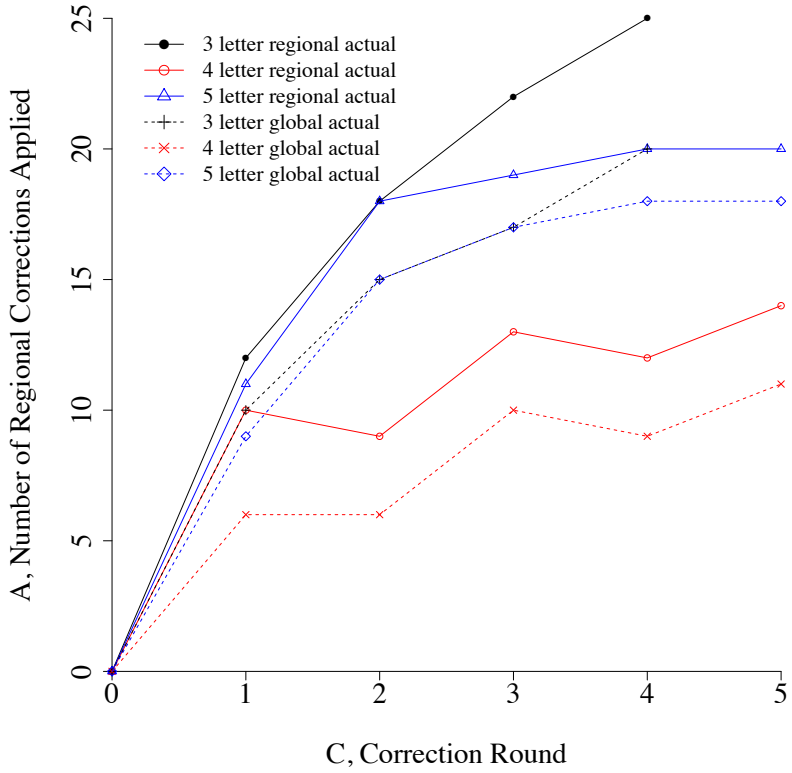


Figure 4.73: Regional corrections applied by correction round for $N = 3, 4, 5$

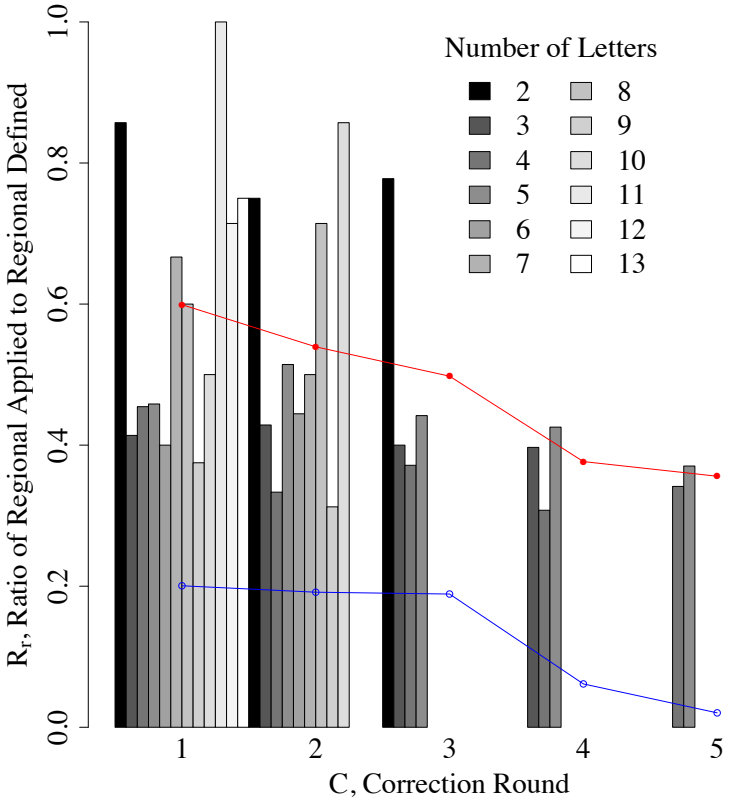


Figure 4.74: Ratio of regional applied to regional defined by correction round

contained misinterpreted words that could not be replaced. Values in parenthesis in the table indicate the corresponding value for the global replacement policy.

Table 4.29: WER regional improvement by correction round

	Correction Round					
N	**0**	**1**	**2**	**3**	**4**	**5**
2	0.28	0.04	0.04	**0**	-	-
3	0.3	0.13 (0.15)	0.15 (0.18)	0.09 (0.14)	**0.01** (**0.06**)	-
4	0.22	0.05 (0.09)	0.08 (0.11)	0.05 (0.08)	0.02 (0.05)	**0** (**0.04**)
5	0.24	0.12 (0.14)	0.09 (0.12)	0.05 (0.07)	0.09 (0.11)	**0.01** (**0.03**)
6	0.1	0.08	**0**	-	-	-
7	0.06	0.07	**0.01**	-	-	-
8	0.05	0.02	**0**	-	-	-
9	0.08	0.8 (0.09)	**0** (**0.01**)	-	-	-
10	0.04	0.03	**0**	-	-	-
11	0.02	**0**	-	-	-	-
12	0.07	**0** (**0.02**)	-	-	-	-
13	0.06	**0** (**0.01**)	-	-	-	-

It should be noted although it is theoretically possible to reduce the WER to zero for all letter categories given sufficient e_i'' definitions with associated tree depth, the following word length categories were not reduced to zero:

- $N = 3$: One additional round of correction would be required to reduce WER to zero (which was not conducted) as round 5 would have included the definition of a homophonic regional rule, e_{10}'' (son→sun). Testing during global replacement would not have improved the WER as this rule can only be defined for regional replacement (similarly for e_5'', sun→son);
- $N = 5$: The final correction round includes a term (*again*) that does not have any recognition value (Google Home consistently had no valid response). A correction rule, e_5'', ^$ →again) could be defined for this purpose, but it possibly far too general for inclusion;
- $N = 7$: Similar to $N = 5$, *Correction Round* 2 has a term (*another*) with no valid recognition status. Due to the generality of the required regional rule, e_1'', ^$ →another), it was not defined and thus this word length category would not benefit from a regional replacement policy in this case.

Figure 4.75 shows the effect of regional replacement on the WER for each word length by each correction round – this is a visualisation of Table 4.29.

The word length category data ($N = 3, 4, 5, 9, 12, 13$) from Table 4.29 that benefit from regional replacement are depicted visually in Figures 4.76 and 4.77. Here the corresponding WER values under a global replacement policy are also shown by the coloured line plots.

The overall effect of the regional policy is also included in Table 4.30, where the WER is averaged across all word length categories for each of the five correction rounds, and by the fifth round the WER is reduced to 0.37×10^{-2}, a 99.19% reduction compared to the initial WER (where no correction has occurred). The mean and standard deviation values in parentheses in Table 4.30 are the values from the global correction policy evaluation data (Table 4.26) for comparison.

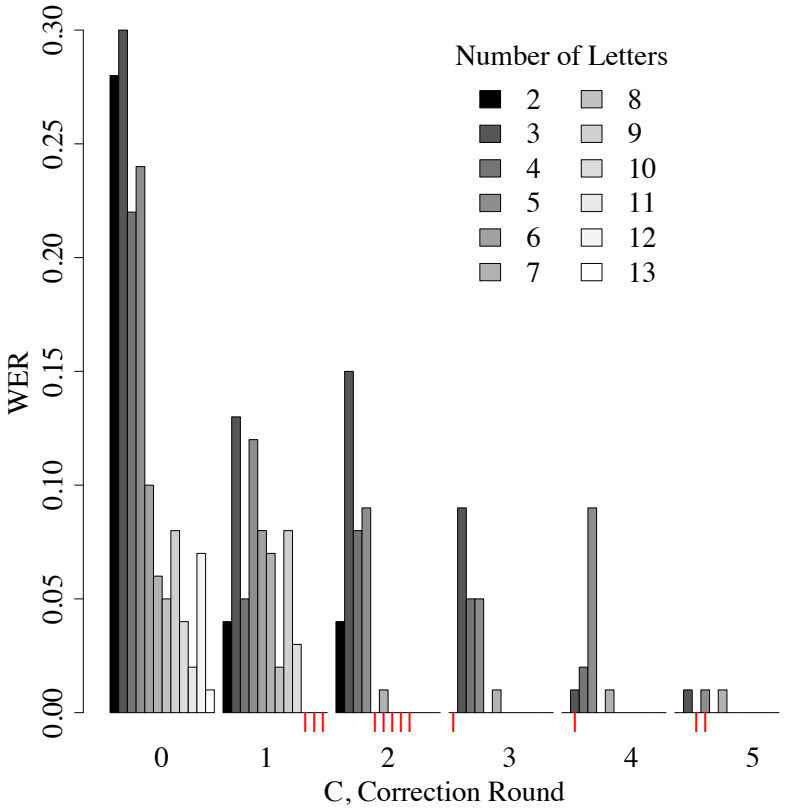


Figure 4.75: WER regional improvement by letter count across correction rounds

Table 4.30: Regional WER improvement by correction round

	Google Home	
Correction round	$\boldsymbol{\mu}(\times 10^{-2})$	$\boldsymbol{\sigma}(\times 10^{-2})$
0 (no correction)	12.64	10.24
1	5.28 (6.60)	4.38 (4.95)
2	3.20 (4.20)	4.91 (6.04)
3	1.78 (2.87)	2.95 (4.46)
4	1.20 (2.28)	2.55 (3.40)
5	0.37 (1.53)	5.53 (1.92)
WER Improvement (0 to 5)	99.19%	

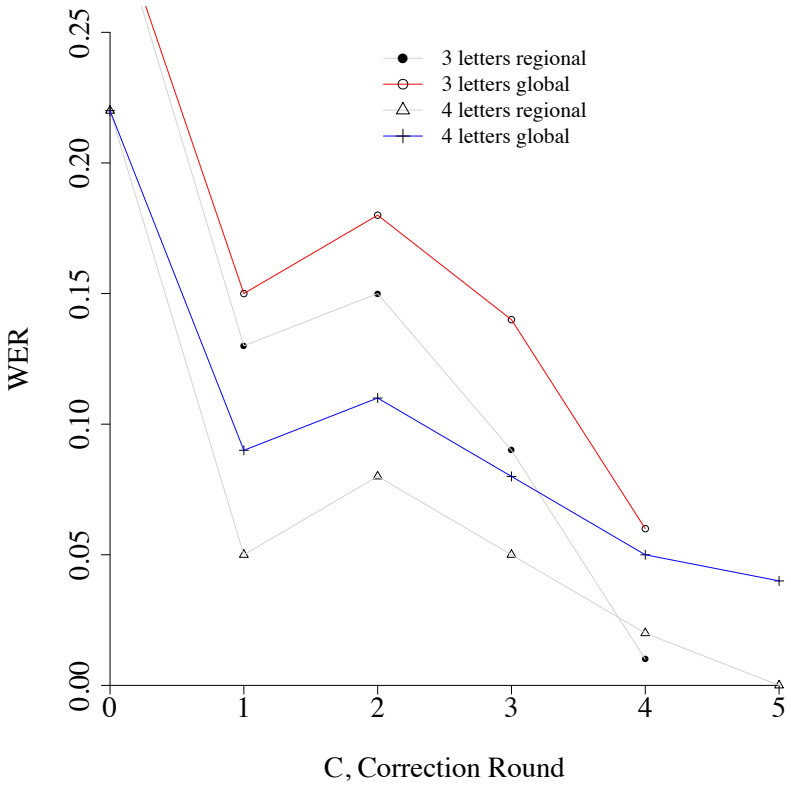


Figure 4.76: WER improvement (regional) over 5 rounds for $N = 3, 4$

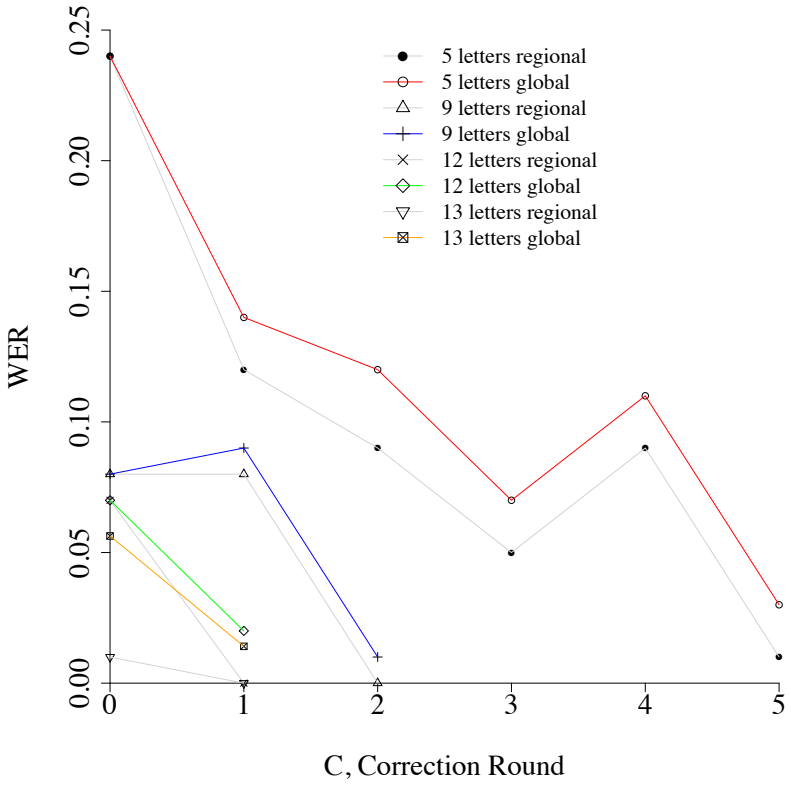


Figure 4.77: WER improvement (regional) over 5 rounds for $N = 5, 9, 12, 13$

4.3.6.3 C-MCRDR WER Improvement Summary

Referring to Table 4.17 in Section 4.3.3, the initial mean measurement of the WER for Google Home (before correction) by letter count for a human speaker was found to be 0.082 and 0.0678 for sentences (see Table 4.21 in Section 4.3.5). These values are very close to those as reported by Këpuska and Bohouta (Këpuska and Bohouta, 2017) of 0.09, and a value of 0.0497 reported by Protalinski (Protalinski, 2017).

A global word correction policy was introduced and applied to correct ASR word misclassification. When applied to the Google Home with a human speaker, for example, with 4 letter words chosen from ranked word corpora, there was a 81.8% reduction in the WER from 0.22 to 0.04 after five rounds of correction acquisition. For all word lengths (with length defined as N), a maximum of five rounds of correction definitions were required (for the $N = 4, 5$ letter word categories). For $N = 6 \ldots 10$, two rounds were needed, and for $N = 11 \ldots 13$ only one round of correction definitions was needed. In terms of the number of global corrections defined by word length category, the greatest number of definitions required occurred for $N = 3$ (53) followed by $N = 5$ (50), and at the other end of the scale, $N = 11$ required the least (2) followed by $N = 13$ (4). As would be expected, the number of new definitions required decreased with each correction round, and for $N = 3, 5$ the data was statistically fitted to logarithmic decreases by correction round. Defining the *effort* of global correction policy rule creation as the ratio between rules being applied during evaluation and rules defined, yields a mean effort (μ_G) of 0.507 - this implies nearly half of all rules defined are not applied (on average) across all correction rounds – this can occur when a rule is defined to correct an ASR error that does not reoccur in later rounds of correction.

Seven of the twelve datasets ($N = 3, 4, 5, 7, 9, 12, 13$) did not have their WER reduced to zero after correction due to the fact the data source included words that, when misinterpreted, are also present in the dataset. As a global correction policy was used, these words cannot be replaced. A prime example is alternate spelling – for example, the $N = 12$ corpora included both *organisation* and *organization*. Google Home recognises *organization* as *organisation*, yet a rule to correct this (*organisation*→*organization*) cannot be defined as this would be replacing a correct source word. This would be overcome if a regional correction policy were used where, in some context, the alternative spelling was needed. Three of the word length categories ($N = 3, 4, 5$) exhibited a statistically significant fitted model of logarithmic decrease in the WER value across the five rounds of correction. Taking the mean WER for all word length categories by correction round, an initial WER prior to correction (and disallowing homophones)

was found to be 0.1264, and in the fifth round of correction, the mean WER was reduced to 0.0153, an 87.9% reduction.

Although a regional correction policy was not implemented in the test environment, reevaluation of the global correction policy evaluation data was conducted to determine the regional rule effects – this was possible as the global correction evaluation data was tagged to indicate rules that could be defined in a regional context but could not be included in a global context. The flow on effects of regional rule definition and subsequent application could then be ascertained. Continuing the example for $N = 4$ from the start of this section, there would be a 100% reduction in the WER from 0.22 to 0 after five rounds of correction if a regional correction policy was used. In terms of the number of regional corrections defined, the greatest number of definitions required would occur again for $N = 3$ (63) followed by $N = 5$ (54), and at the other end of the scale, $N = 11$ would require the least (2) followed by $N = 13$ (6). The mean *effort* required improves in contrast to the global correction policy, with $\mu_r = 0.529$. Although not evaluated in the testing environment with the data corpora used, the *effort* ratio could actually exceed 1.0 if a defined rule was applied to more than one identified ASR transcription error, for example, a more generalised regular expression might be defined for multiple correction matches rather than a single word-for-word match. This would be a desirable outcome as increasing the *effort* ratio means less actual effort for the conversational author in a production system.

All word length category datasets would have a reduction in WER to zero under a regional correction policy except for $N = 3, 5, 7$. $N = 5, 7$ both included a single term with no ASR response (which could only be potentially corrected with a too-general corrective rule), and $N = 3$ would have required an addition (fifth) correction round for one term that was not actually conducted in the global correction policy evaluation. Taking the mean WER for all word length categories by correction round, and with the same initial WER prior to correction of 0.12640, after five rounds of correction, the mean WER would be reduced to 0.0037, a 99.19% reduction. This is a significant improvement compared to the global correction policy final WER of 0.0153.

Chapter 5

Conclusions and Future Work

> "*I am rarely happier than when spending an entire day programming my computer to perform automatically a task that it would otherwise take me a good ten seconds to do by hand.*"
> – Douglas Adams (Adams and Carwardine, 1990)

This chapter contains the following sections:

5.1 Introduction

The ability to interact with emerging digital services through conversation and modify Internet of Things (IoT) device behaviour by imparting human knowledge conversationally is a challenge that needs to be addressed. The objective of this thesis was to start to address this challenge by

defining and evaluating a rule-based conversational agent system that naturally retains *conversational context* and that can be easily maintained by conversational authors who are experts in their own domains.

The purpose of this thesis is to contribute to the body of knowledge surrounding rule-based knowledge-base systems as applied to conversational agent systems by addressing the core research question:

> **Core RQ:** *How can a human-authored rule-based knowledge-base system be defined and applied to create a conversational agent system that can achieve high levels of system performance without relying on complex scripting and programming skills or knowledge of formal grammatical syntax to specify the conversational knowledge?*

To answer this core question, the research results are now framed in the context of answering the research questions posited in Section 1.2.1. Table 5.1 summarises the answers to each sub-research question. More detail on methodologies can be found in Chapter 3 and evaluation results and discussion in Chapter 4.

Table 5.1: Research question result summary

RQ	Summary
SRQ1	*How can conversational context be maintained in a human-authored rule-based conversational agent system?*
Answer	Conversational (topical) context is maintained by adopting a stack-based approach to store previous conversational-based inference requests to the C-MCRDR knowledge-base. Inference requests start with rules that were satisfied from the top-of-stack frame - see **Section 3.3.1 Inference Modification – Topical Conversational Context** that details the method, and in particular, see Table 3.3 for a description of the enhanced C-MCRDR inference algorithm. ***Evaluation*** A stack-based approach was evaluated via the C-MCRDR CA system, that showed for example, that conversational depth in the C-MCRDR knowledge-base maintained an average rule depth of 4. A satisfied rule depth greater than one is indicative of topical context being retained and used between inference requests in each participant session in the test domain. This is because the majority of each participant's inference requests only satisfied one rule per request. Please see **Section 4.2.4 Rules Satisfied During Evaluation** for the evaluation results.

Continued on next page

Table 5.1 – *Continued from previous page*

RQ	Summary
SRQ2	*What is a pattern-matching approach that does not require complex scripting and programming skills or knowledge of formal grammatical syntax that can provide high levels of system performance?*
Answer	A lexical paraphrasing approach, where the conversational author can establish pattern-matching key terms and associated lexical paraphrases, allows conversational knowledge aligned to a target pedagogical domain to be quickly acquired - see **Section 3.3.2 Dictionary/Lexical Paraphrase Approach**. ***Evaluation*** The lexical paraphrasing approach was evaluated to measure if the approach yielded a conversational agent that provided correct and appropriate responses through analysis of the interaction results of participant users of the C-MCRDR CA system. Analysis of the C-MCRDR CA system's log files containing the inference request and responses data determined the average response rate where responses were categorised based on manual inspection of the user's intent. This analysis demonstrated the system responded correctly (and appropriately) in 80.3% of its responses to participant inference requests in the trial. Despite the fact that the C-MCRDR CA system's knowledge-base was relatively shallow, and the domain-specific rules themselves were not complex (See Appendix A), the measurement of the system's correct response rate showed the lexical paraphrasing approach achieved excellent results in the trial - see **Section 4.2 C-MCRDR CA System Evaluation**. Integrated feedback mechanisms measured user satisfaction from two perspectives – the overall perception of system performance and the perception of individual responses to inference requests. Analysis of the users' participation feedback data demonstrated overall system performance feedback was very positive - the highest frequency scores (on a 5-point Likert scale, 1=*I am very dissatisfied* …5=*I am very satisfied*) were ratings at levels 4 and 5 (61.9% combined). For individual system response feedback, 46.46% of feedback was at score 5, *My answer is exactly what you are seeking*, and if score 4 feedback *My answer is correct but some information is missing* is included, positive satisfaction rises to 53.55% of response feedback – see **Section 4.2.10 User Acceptance (Feedback)**.

Continued on next page

Table 5.1 – *Continued from previous page*

RQ	Summary
SRQ3	*What augmentations to the MCRDR rule-based KBS methodology will produce conversational agent systems that can achieve high levels of system performance?*
Answer	The required augmentations to the MCRDR KBS methodology for conversational agents were determined by considering a targeted pedagogical domain. This included determining what conversational expressiveness was required by domain users to achieve their conversational intent that could then be specified by the conversational author with limited knowledge of programming or grammatical syntax in the C-MCRDR knowledge-acquisition process. These augmentations were then generalised to be domain-agnostic in order to produce a tool that can be applied to alternative domains. Conversation context retention is achieved via the inference-stacking approach of C-MCRDR. Other augmentations were also included to achieve brittleness mitigation, NLIDB, and initial ASR error correction: • definition of individual intra- and inter-dialog context (see **Section 3.3.4 Context Variables**); • inclusion of this context as potential attributes in rule antecedents (although implemented, this was not evaluated in the C-MCRDR CA system) (see **Section 3.3.5 C-MCRDR Inference**); • inclusion of this context as potential output in rule consequents (see **Section 3.3.4.1 Context Variable Definition**); • inclusion of this context as potential binding data in generalised database queries (see **Section 3.3.6.1 Database Querying**); • decreasing rule maintenance and consequently authoring overheads for the conversational author by the inclusion of generalised database queries in the post-inference stage - see **Sections 3.3.6 Post-Inference** and **3.3.6.1 Database Querying**; • mitigating brittleness by incorporating a topical look-ahead phase of inference - see **Section 3.3.8 Brittleness Mitigation**; • considering speech issues, including speech-to-text input correction, and text-to-speech pronunciation correction - see **Section 3.3.9 Speech Considerations**.

Continued on next page

Table 5.1 – *Continued from previous page*

RQ	Summary
SRQ3 Answer *Continued*	***Evaluation*** The C-MCRDR CA knowledge-base was populated by the thesis author, with careful consideration to ensure all knowledge acquisition was conducted with an emphasis on minimal scripting technical expertise requirements, minimal knowledge of formal grammatical methods and overhead. Complete domain ontological terminology coverage was achieved with a minimum number of rules (see **Table A.3** in **Appendix A**). The average correct response rate at 80.3% reflects a successful conversational agent system produced with minimal overhead for the conversational author. The inclusion of generalised database queries in the post-inference phase to reduce rule bloat was demonstrated successfully in **Section 4.2.3 Rule-count Reduction** and **Section 4.2.5 Effective Query Execution**, which also significantly reduces the rule (knowledge) acquisition overhead for the conversational agent author. This also had the significant effect of extending the conversational agent to also be a **Natural Language Interfaces to Databases** (NLIDB) agent. Initial speech considerations were evaluated in **Sections 4.2.9 Automatic Speech Recognition (ASR) errors**, **4.2.9.1 Speech-to-text Correction** and **4.2.9.2 Text-to-speech Correction**. In addition to the analysis conducted within this doctoral study the C-MCRDR CA system has been successfully leveraged in two other recent research publications where the conversational agent knowledge acquisition was conducted by non-expert conversational authors. • Clinical training via a 3D dialog simulation (Yang et al., 2019): *…a 3D interactive dialogue-based training and assessment system that supports the detailed development of clinical trial competency for medical students in a distributed virtual environment* • Task-oriented language assistant for improving English as a second language (Gu et al., 2019): … [a system] *to improve ESL students English speaking skills, while at the same time providing assistive information for international students studying in foreign universities*

Continued on next page

Table 5.1 – *Continued from previous page*

RQ	Summary
SRQ4	a) *What is the best current market-leading Intelligent Personal Assistant (IPA) to leverage as a speech component of a conversational agent system in terms of ASR performance that allows the default vendor-defined conversational agent to be supplanted?*; and b) *How can the IPA's associated ASR errors when coupled to a human-authored rule-based KBS be corrected?*
Answer	Only two *market-leading* IPA devices were tested due to the constraints that a device must include an accessible application programming interface (API), and the API's workflow must support the exposure of the raw speech-to-text ASR result (see **Chapter 3.4 Intelligent Personal Agent Methodology**). For example, Apple's Siri (via the Apple Homepod IPA) (Apple, 2018) did not honour the constraints and thus was not tested. Evaluation showed the Google Home was the best-performing device. When coupled to the C-MCRDR CA system, the best-performing device's raw speech-to-text ASR result is available for inspection and/or modification. Two methods to correct ASR transcription errors were defined (see **Section 3.4.2 Recognition Performance Improvement**) taking into account the context of where the error occurs: 1. *Global ASR correction* (**Section 3.4.2.1 Global ASR Correction**) also takes a similar, but simpler lexical paraphrasing approach to that employed by the CA system's knowledge acquisition process, but context is not considered; and 2. *Regional ASR correction* (**Section 3.4.2.2 Regional ASR Correction**) does consider context. Note that although defined, this was not implemented in the C-MCRDR CA system (see **Chapter 5 Conclusions and Future Work**). An introspective evaluation of regional ASR correction was conducted through re-evaluation of the global ASR correction evaluation results. ***Evaluation*** Testing consisted of logging the ASR results of spoken utterances from a well-established data corpora, as well as sentences generated via a Markov chain from a copyright-free novel. Measuring the Word Error Rate (WER) for the Google Home and Amazon Echo revealed that the WER for isolated words spoken by the thesis author was lower for the Google Home (8.20×10^{-2}) compared to the Amazon Echo's WER of 27.16×10^{-2}. Across each test category the WER for the Google Home was significantly lower (better) than that of the Amazon Echo.

Continued on next page

Table 5.1 – *Continued from previous page*

RQ	Summary
SRQ4 Answer ***Continued***	Please see **Section 4.3 Intelligent Personal Assistant Evaluation**. • The global ASR correction method was evaluated for the Google Home with a human speaker, and the WER was improved by 87.9%, from 12.64×10^{-2} to 1.53×10^{-2} (see **Section 4.3.6.1 Global ASR Correction**). • The regional ASR correction method was evaluated through introspection for the Google Home with a human speaker, and the WER was improved from 12.64×10^{-2} to 0.37×10^{-2}, a 99.19% reduction (see **Section 4.3.6.2 Regional ASR Correction**).

Having reached a conclusion for each sub-research question, an answer can now be formulated for the core research question:

> **Core RQ:** *How can a human-authored rule-based knowledge-base system be defined and applied to create a conversational agent system that can achieve high levels of system performance without relying on complex scripting and programming skills or knowledge of formal grammatical syntax to specify the conversational knowledge?*

Answer

This doctoral study has shown that by augmenting the MCRDR KBS to retain contextual information, and utilizing a lexical paraphrasing pattern-matching approach to reduce the cognitive load on the conversational author, and coupling with an ASR error-corrected IPA device as an interface produces a CA system that has high-levels of system performance that also does not require complex scripting and programming skills or knowledge of formal grammatical syntax by the conversational author.

5.2 Significant Findings

This section summarises the significant findings in relation to the literature to identify the significant contributions to knowledge-based conversational agent systems. Limitations related to the significant findings are discussed in Section 5.2.7.

5.2.1 Maintaining Conversational Context

Finding 1: *C-MCRDR can facilitate an effective, efficient way of maintaining knowledge base systems by acquiring and classifying knowledge incrementally and unlike MCRDR, it maintains conversation context, a factor that assists with allowing utterance classification to change over time as a conversation progresses.*

C-MCRDR has implicit retention of topical conversational context by adopting a stack-based modification to MCRDR's inference mechanism (Kang, 1995). Conversational topical context is thus retained "for free" without any explicit rule or algorithmic effort on the conversational author's behalf. Additional context between dialog utterances in the C-MCRDR CA system is maintained through reference to context-based variable definition and assignment via simple regular expression pattern-matching. C-MCRDR context variables can be considered a modern form of knowledge representation like the relatively older concept of *terminals* in Minsky's Frame structure (Minksy, 1975). The C-MCRDR CA system, which has the novel C-MCRDR as the core KBS methodology, was applied to a pedagogical domain to answer undergraduate students' questions pertaining to their enrolled course-related content (Herbert and Kang, 2018). Analysis showed a very high level of system performance in terms of how it responded to user questions – 80.3% of participant requests received *appropriate* responses.

5.2.2 Rule Count Reduction

Finding 2: *Substantial rule-count reduction can be achieved by C-MCRDR (in comparison to standard MCRDR) due to post-inference deferred classifications that include database querying expressions.*

C-MCRDR demonstrated substantial rule-count reduction in comparison to standard MCRDR (Kang, 1995), due to post-inference deferred classifications that include database querying expressions (bound by relevant context variables). This is a natural form of implicit compression (Suryanto et al., 1999) as it allows the conversational author to think in terms of generalising conversational responses and thus reduces the number of rules needed overall to achieve the same classification results compared to MCRDR. In the evaluated domain, the C-MCRDR

approach had a 96.9% reduction in rule count compared to a MCRDR approach that would not have conversational context retention and post-inference database querying in rule consequents.

5.2.3 NLIDB Framework

Finding 3: *A natural language interfaces to databases (NLIDB) framework can be engendered by C-MCRDR by way of post-inference query binding and a pattern-matched NL interface with the advantages of knowledge acquisition and maintenance through the ripple-down methodology.*

The C-MCRDR CA system defines a methodology for incremental knowledge acquisition and knowledge maintenance that is based on the original RDR approach (Compton and Jansen, 1988). The C-MCRDR CA conversational author can define generalised database queries through a GUI-guided process. These queries may also include reference to maintained context through context variables that will produce more specificity in query results. The overall result then allows the C-MCRDR CA system to provide an NL interface to database querying in a similar manner to other NLIDB systems (Bais et al., 2016; Nguyen et al., 2016). The conversational author may build a series of NLIDB queries incrementally through the simplicity of the pattern-matching approach adopted by the C-MCRDR CA system. In the test domain, 44 queries were defined (see Appendix A, Table A.9) and they were associated with 34 rules defined in the C-MCRDR CA knowledge-base.

5.2.4 Brittleness Mitigation

Finding 4: *Brittleness can be mitigated by C-MCRDR due to the incremental approach to knowledge acquisition, the use of paraphrasal terms in pattern matching, and paraphrasal look-ahead prompting.*

C-MCRDR mitigates brittleness through the dynamic maintenance of acquiring new knowledge in the form of rules incrementally where knowledge was previously lacking (Compton et al., 2006). An additional form of dynamic maintenance also includes the update of lexical terms. As the C-MCRDR CA system uses paraphrasal or lexical pattern matching (Bradeško and Mladenić, 2012), the matched lexical terms dictionary may be updated which can increase paraphrasal coverage at any time, either during or outside of formal knowledge acquisition. This is in contrast to *static* slot-filling definitions of, for example, IPA methodologies (Amazon, 2018; Google, 2018*a*).

The second main brittleness mitigation technique defined by the C-MCRDR CA system is through constrained utterance *suggestion* (by rule lookahead) based on the current topical

context. When an utterance is not recognised or when a user explicitly requests help, the C-MCRDR CA system provides a look-ahead suggestion. C-MCRDR rules are evaluated from the current context (a stack frame) to determine which attributes will satisfy the look-ahead rule's antecedents - in effect an inference request that is then *not stacked*. NLIDB approaches that iteratively ask a user to refine their query until it is recognised and correct (Li et al., 2005; Li and Jagadish, 2014*b*,*a*; Smith et al., 2014) could be interpreted as a feedback system. However, such systems *refine* a query and as a consequence they do not mitigate brittleness through suggestive indicators of the system's knowledge.

In the test domain, brittleness mitigation was found to be an important and probable factor in approximately 71% of all non-root rule (*I don't understand*) inference responses.

5.2.5 IPA ASR Performance

Finding 5: *The Google Home values for the isolated word and phrasal recognition word error rate (WER) measurements were significantly lower than the Amazon Echo.*

No studies found to date comprehensively evaluate and measure the isolated word or sentence-level word error rate (WER) of contemporary IPA devices. Nor do they examine the relationships (if any) of word and sentence lengths with the WER, or word ranking by a spoken corpora on the WER. Such evaluations are performed within this doctoral study.

Normally, ASR errors are hidden behind the probability matching of utterance to intent by *slot filling* (Google, 2018*a*; Amazon, 2018; Hoy, 2018; de Barcelos Silva et al., 2020). When the IPA device's ASR transcription errors are exposed and measured by the WER, the Google Home values for the isolated word and phrasal recognition rates were found to differ significantly to the Amazon Echo. For example, the Google Home's average WER was $\approx 70\%$ lower compared to the Amazon Echo for isolated words when the source utterances came from a human.

5.2.6 ASR Error Correction

Finding 6: *Two error correction schemes have been defined that can be applied to IPAs when coupled to a C-MCRDR CA system to significantly improve the average WER across all datasets.*

Two approaches for a conversational author to correct ASR transcription errors to match the domain-specific ontology were defined – *global* and *regional* ASR correction. These are forms of manual, supervised correction editing (Ringger and Allen, 1996; Mangu and Padmanabhan, 2001). To retain the philosophy of not requiring the conversational author to possess technical expertise in programming or scripting both approaches also adopt a pattern-matching rule-based

methodology. The global ASR correction scheme was implemented, while the implementation of the regional ASR correction scheme has been deferred to future work. Careful considerations for the regional scheme implementation will be needed in order to avoid excessive reliance or exposure to the decision tree structure (which would violate the RDR philosophy of defining knowledge in the context in which it arises). The global correction scheme achieved correct recognition rates of 100% in five of the data sets evaluated, and across all datasets the average WER was decreased by 81.34%. Introspective analysis indicated the regional correction scheme would improve on these results, it would achieve a mean WER reduction of 99.19%.

5.2.7 Limitations

Table 5.2 details the limitations for this doctoral study. The first four limitations detailed are applicable to Findings 1 – 4, whereas the latter two limitations are applicable to Findings 5 and 6.

Table 5.2: Research limitations

Issue	**Comment**
Participants	Participants in the C-MCRDR CA system evaluation were all undergraduate students in ICT at different levels of their degree. The technical competency of such participants is assumed therefore to be higher than the average "lay" person, and such participants would be used to using browser-based clients and perhaps formulating spoken or textual queries with a greater degree of formality in comparison, which may be a form of selection bias. The pedagogical domain assessed however was directly applicable to the participant's experience and environment as it detailed unit and course information in response to constrained NL queries.
Target domain	The limited evaluation of the C-MCRDR CA system through one domain is acknowledged. Evaluation of user acceptance and system performance in other domain-specific contexts will need to be conducted. For example, the technical competency of participants mentioned above would need to be verified that it does not bias the overall user acceptance and system performance results, and that the validation of the approach is not limited to one specific domain.

Continued on next page

Table 5.2 – *Continued from previous page*

Issue	Commment
Domain expert	The thesis author conducted all knowledge acquisition for the C-MCRDR CA system definition and subsequent evaluation as the author was the domain expert for the pedagogical domain assessed (i.e. as an academic for the *unit outline* document content as well as previously being a developer for a legacy online outline documentation system. However, the legacy development skills were very carefully *not employed* in the KA process). It must be noted the overall philosophy of the simplicity of the interface and non-reliance on scripting or programming was a primary focus throughout in order to ensure the process was not biased. Assuming the domain expert role, conversational knowledge was acquired by considering common questions that are asked by students in the domain, typically by association with key terms from unit outline nomenclature, which are then also naturally mapped by the *in situ* legacy referential database schema. Independent validation of the approach was achieved through two other studies as detailed in Table 5.1 that used the C-MCRDR CA system with completely initialised (i.e. empty) knowledge-bases and independent conversational authors.
MCRDR	It must be noted the test domain evaluation results primarily consisted of single-response inference classifications. Although the implementation, augmentations and underlying methodology supports multiple classifications, C-MCRDR CA system evaluation study participants typically phrased the majority of their questions that (correctly) only resulted in a single classification each time. Further work may include evaluation that guide the participants in the formulation of natural language queries that result in multiple classification responses.
Validation	Although implemented in the development code, reference to context variable values as rule antecedents has not been evaluated or assessed in terms of the effect to standard MCRDR validation (see Section 2.2.5.4). In a similar manner to the effect of introducing cyclic dependency like the Round Rules in MCRRR (Bindoff, 2010), careful consideration of this augmentation needs to be conducted.

Continued on next page

Table 5.2 – *Continued from previous page*

Issue	Commment
IPA hardware	The WER measurements of both IPA devices evaluated, although conducted under identical physical environments (acoustics, speaker, volume etcetera) do not take into account any variations in the physical hardware itself, for example, the number of and quality of microphones, echo cancellation, latency in processing and so on. The assumptions made in essence are the associated IPA agents (Google Assistant and Amazon Alexa) are being independently evaluated for their ASR performance, but the quality and attributes of the hardware would most likely be a factor. As one of the research question being answered was to ascertain which was the best *device*, hardware differences in that case should not be a factor when considering the performance in its entirety.
IPA source	Variations in speaker (for example, accent, gender and age) were not evaluated as to their effect on WER. The measurements of WER were primarily conducted with a single human speaker (the thesis author) and a computer generated voice for comparison, but it is acknowledged this is a very limited sample size. Despite this fact, the lack of variability did result in dramatic differences in measured performance between both IPA devices evaluated. Further studies may include both a wider variation in speakers, and a larger set of IPA devices (for example, different devices from the same or different vendors that have the same associated agent, such as the Google Nest Hub, or Sonos One). In fact, devices that can support different agents (such as the Sonos One (*Sonos Wireless Speakers*, 2020)) may be ideal for evaluation as the hardware is identical and only the back-end agent is changed through configuration settings. As less restrictive APIs become available and hardware is released, further agent-related IPAs may be evaluated (such as Siri (Apple, 2018), Cortona (Microsoft, 2018) and Bixby (*Bixby*, 2020))

5.3 Future Work

The thesis research touches on many fields that are candidates for extension through future work. The main extensions identified are detailed in Table 5.3.

Table 5.3: Future Work

Topic	Comment
Target domain	As mentioned in Table 5.2, evaluation of system performance in other domain-specific contexts can be conducted to further strengthen the validity of the approach is not limited to one specific domain.
Domain expert	Further evaluation of the approach (with the emphasis on the limited programming/scripting expertise of the conversational author (the domain expert)) with a wider, more varied set of domain authors in different domains may be conducted. Such a study may include a qualitative evaluation by conversational author participants to determine their perceptions of the ease of knowledge acquisition and maintenance, coupled with the resultant system performance in their domain of expertise.
Robot control	A much larger extension of the work could include investigating the application of C-MCRDR to create a system to facilitate non-programmers to behaviourally train and control autonomous systems. To extend the system beyond mere teleoperation, the extension can include defining a form of hierarchical behavioural abstraction that can be maintained through a C-MCRDR CA system to allow differing degrees of autonomy, with the conversational author able to refine and/or enhance defined behaviours through knowledge acquisition. Another possibility is that an autonomous system's integrated, on-board sensor feedback can be presented to a suitably-augmented MCRDR system for classification and subsequent action directives as followup. Leveraging existing robotic platforms such as the Robot Operating System (ROS) (Robotis, 2018) would drastically reduce the development work required.
Performance and Brittleness	The C-MCRDR CA system's performance was evaluated with brittleness mitigation enabled (look-ahead hinting, Section 3.3.8). Evaluation (Section 4.2.6), showed that this had a significant affect on the resultant system performance. Investigating performance with this specific mitigation technique disabled could be evaluated through an additional multi-participant study.

Continued on next page

Table 5.3 – *Continued from previous page*

Topic	Commment
Ontology	Expanding the ontological coverage of the approach can be investigated, through developmental integration of the system's dictionary with, for example, OpenCyc (*OpenCyc*, 2018) or WordNet (Miller, 1995).
NLIDB	The NLIDB aspects of the research are associated with manual creation of both the database queries and the pattern-matching terms for the natural language query. A collaborative project (that was not completed) involved automatically examining the domain-specific database schemas and generating referential context that is then available to the conversational author. This, coupled with automatic query generation such as that performed by the SEEKER system (Smith et al., 2014) and question generation processes like the cQA system (Figueroa, 2017) are promising avenues to investigate to ease the burden of the process for the conversational author.
String matching	Investigating alternative string matching techniques (such as n-grams, but constrained by the overall philosophy of low technical overheads for the conversation author) as mentioned in Section 2.3.3 may result in significant increases in system performance and a reduction in brittleness. An evaluation of this effect may be considered.
C-MCRDR inference	The current developmental code includes implementation to allow rule antecedents to include context variable referral as attributes, direct database access as a triggered context variable action (which allow maintenance and manipulation of persistent state) and a meta-rule indicating inference results should not be stacked (which is currently used for the brittleness mitigation lookahead method). These modifications have not been directly evaluated nor have they been formally defined as to their affect on inference, so a formal evaluation may be warranted.
Regional correction	A regional correction policy for ASR transcription errors was evaluated in Section 4.3.6.2 for its effect but it was not implemented in the developmental code. Implementation will require careful interface considerations in order to potentially abstract rule regions in the decision tree from the conversational author in the normal process of knowledge acquisition.

Continued on next page

Table 5.3 – *Continued from previous page*

Topic	Commment
Maintenance effort	The C-MCRDR CA system's performance was evaluated in Section 4.2.8, however, the maintenance effort required to *increase* performance in terms of brittleness reduction and correctness of system response was not evaluated. The pedagogical rule-base for evaluation was not altered throughout evaluation, and so this can be the subject of further investigation that is also coupled to evaluating the qualitative perception of effort of different domain experts in authoring conversational knowledge.

5.4 Conclusion

This doctoral study has presented an approach to creating conversational agent (CA) systems through the augmentation of a rule-based knowledge-base system (KBS) methodology called Contextual MCRDR (C-MCRDR), to allow conversational authors to provide the conversational knowledge without the need for complex scripting and programming skills or knowledge of formal grammatical syntax. A CA system based on C-MCRDR was developed and in a pedagogical domain where typical problems associated with knowledge-base systems, such as knowledge-acquisition bottlenecks and brittleness (Compton and Jansen, 1988; Lenat et al., 1985), were addressed through the choice and augmentation of MCRDR. Two market-leading intelligent personal assistants (IPA) were then extensively evaluated for their speech-to-text performance in order to determine which device to adopt as a speech-enabled interface to the C-MCRDR CA system and two post-transcription word-correction schemes were then evaluated as a consequence.

This thesis has made a substantive contribution to the body of knowledge surrounding rule-based KBS as applied to conversational agent systems by:

- Augmenting the MCRDR KBS methodology with contextual considerations to create a new rule-based KBS methodology called Contextual MCRDR (C-MCRDR) that can be utilised within a CA system;
- Developing a CA system based on C-MCRDR that can acquire and apply rule-based conversational knowledge from conversational authors who do not possess complex scripting and programming skills or knowledge of formal grammatical syntax;

- Determining whether the C-MCRDR CA system results in a well-performing system in terms of the correctness of the system's responses to conversational questions in a target domain as evaluated by the system architect and domain user's feedback;
- Determining and evaluating methods to mitigate KBS brittleness through the application of C-MCRDR;
- Evaluating the ASR error rates of market-leading IPA devices and identifying attributes of speech inputs that impact the error rates;
- Determining which market-leading IPA device is the best choice based on its ASR performance to use as a speech interface that allows the default vendor-defined CA to be replaced by the developed C-MCRDR CA; and
- Defining and evaluating methods to correct ASR errors from IPA devices when they are coupled to a C-MCRDR CA system.

REFERENCES

Aamodt, A. and Plaza, E. (1994), 'Case-based reasoning: Foundational issues, methodological variations, and system approaches', *AI communications* **7**(1), 39–59.

Abdul-Kader, S. A. and Woods, J. (2015), 'Survey on chatbot design techniques in speech conversation systems', *International Journal of Advanced Computer Science and Applications* **6**(7).

Abraham, A. (2005), 'Rule-based expert systems', *Handbook of measuring system design* .

AbuShawar, B. and Atwell, E. (2016), 'Usefulness, localizability, humanness, and language-benefit: additional evaluation criteria for natural language dialogue systems', *International Journal of Speech Technology* **19**(2), 373–383.

Adams, D. (1980), *The Restaurant at the End of the Universe*, Pan Books.

Adams, D. and Carwardine, M. (1990), *Last chance to see*, Pan Books.

Adams, P. H. and Martell, C. H. (2008), Topic detection and extraction in chat, *in* 'Semantic Computing, 2008 IEEE International Conference on', IEEE, pp. 581–588.

Adiwardana, D., Luong, M.-T., So, D. R., Hall, J., Fiedel, N., Thoppilan, R., Yang, Z., Kulshreshtha, A., Nemade, G., Lu, Y. et al. (2020), 'Towards a human-like open-domain chatbot', *arXiv preprint arXiv:2001.09977* .

AIML Tutorial (2020). [Accessed 1 June 2020].
URL: *https://www.tutorialspoint.com/aiml*

Alencar, M. and Netto, J. M. (2011), Improving cooperation in virtual learning environments using multi-agent systems and aiml, *in* '2011 Frontiers in Education Conference (FIE)', IEEE, pp. F4C–1.

Amazon (2018), 'Alexa skills kit'. [Accessed 18 Sep 2018].
URL: *https://developer.amazon.com/alexa-skills-kit*

Androutsopoulos, I., Ritchie, G. D. and Thanisch, P. (1995), 'Natural language interfaces to databases – an introduction', *Natural Language Engineering* **1**(1), 29–81.

Androutsopoulou, A., Karacapilidis, N., Loukis, E. and Charalabidis, Y. (2019), 'Transforming the communication between citizens and government through ai-guided chatbots', *Government Information Quarterly* **36**(2), 358–367.

Angell, R. C., Freund, G. E. and Willett, P. (1983), 'Automatic spelling correction using a trigram similarity measure', *Information Processing & Management* **19**(4), 255–261.

Apple (2018), 'Sirikit'. [Accessed 18 Sep 2018].
URL: *https://developer.apple.com/documentation/sirikit*

Ardiny, H., Witwicki, S. and Mondada, F. (2015), 'Are autonomous mobile robots able to take over construction? a review', *International Journal of Robotics, Theory and Applications* **4**(3), 10–21.

Argall, B., Chernova, S., Veloso, M. and Browning, B. (2009), 'A survey of robot learning from demonstration', *Robotics and autonomous systems* **57**(5), 469–483.

Argall, B., Gu, Y., Browning, B. and Veloso, M. (2006), The first segway soccer experience: Towards peer-to-peer human-robot teams, *in* 'Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction', ACM, pp. 321–322.

Arkin, R. C. (1995), Just what is a robot architecture anyway, *in* 'Turing equivalency versus organizing principles, in AAAI Spring Symposium: Lessons Learned from Implemented Software Architectures for Physical Agents'.

Atkeson, C. G. and Schaal, S. (1997*a*), Learning tasks from a single demonstration, *in* 'Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on', Vol. 2, pp. 1706–1712.

Atkeson, C. G. and Schaal, S. (1997*b*), Robot learning from demonstration, *in* 'ICML', Vol. 97, pp. 12–20.

Bachant, J. and McDermott, J. (1984), 'R1 revisited: Four years in the trenches', *AI magazine* **5**(3), 21.

Badham, J. (1983), 'Wargames'. Film, MGM/UA Entertainment, USA.

Bais, H., Machkour, M. and Koutti, L. (2016), Querying database using a universal natural language interface based on machine learning, *in* 'Information Technology for Organizations Development (IT4OD), 2016 International Conference on', IEEE, pp. 1–6.

Bakar, Z. A., Sembok, T. M. T. and Yusoff, M. (2000), 'An evaluation of retrieval effectiveness using spelling-correction and string-similarity matching methods on malay texts', *Journal of the American Society for Information Science* **51**(8), 691–706.

Barabás, P., Kovács, L. and Vircikova, M. (2012), Robot controlling in natural language, *in* 'Cognitive Infocommunications (CogInfoCom), 2012 IEEE 3rd International Conference on', IEEE, pp. 181–186.

Bassil, Y. and Semaan, P. (2012), 'Asr context-sensitive error correction based on microsoft n-gram dataset', *arXiv preprint arXiv:1203.5262* .

Batarseh, F. A. and Gonzalez, A. J. (2015), 'Validation of knowledge-based systems: a reassessment of the field', *Artificial Intelligence Review* **43**(4), 485–500.

Beydoun, G. and Hoffmann, A. (1997), Acquisition of search knowledge, *in* 'International Conference on Knowledge Engineering and Knowledge Management', Springer, pp. 1–16.

Beydoun, G. and Hoffmann, A. (2001), 'Theoretical basis for hierarchical incremental knowledge acquisition', *International Journal of Human-Computer Studies* **54**(3), 407–452.

Biermann, J. (1998), Hades—a knowledge-based system for message interpretation and situation determination, *in* 'International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems', Springer, pp. 707–716.

Bindoff, I. K. (2010), Multiple Classification Ripple Round Rules: Classifications as Conditions, Thesis, Computing.

Bindoff, I. and Kang, B. H. (2011), Applying multiple classification ripple round rules to a complex configuration task, *in* 'Australasian Joint Conference on Artificial Intelligence', Springer, pp. 481–490.

Bindoff, I., Tenni, P., Kang, B. H. and Peterson, G. (2006), Intelligent decision support for medication review, *in* 'Pacific Rim Knowledge Acquisition Workshop', Springer, pp. 120–131.

Bisani, M. and Ney, H. (2004), Bootstrap estimates for confidence intervals in asr performance evaluation, *in* '2004 IEEE International Conference on Acoustics, Speech, and Signal Processing', Vol. 1, pp. I–409.

Bixby (2020). [Accessed 8 June 2020].
URL: *https://www.samsung.com/global/galaxy/apps/bixby/*

Bonner, S. and Shin, K. G. (1982), 'A comparative study of robot languages', *Computer* **15**(12), 82–96.

Bradeško, L. and Mladenić, D. (2012), A survey of chatbot systems through a loebner prize competition, *in* 'Proceedings of Slovenian Language Technologies Society Eighth Conference of Language Technologies', pp. 34–37.

Brooks, R. (1986), 'A robust layered control system for a mobile robot', *IEEE journal of robotics and automation* **2**(1), 14–23.

Brooks, R. A. (1990), 'Elephants don't play chess', *Robotics and autonomous systems* **6**(1), 3–15.

Brooks, R. A. (1991), 'Intelligence without representation', *Artificial intelligence* **47**(1-3), 139–159.

Browning, B., Xu, L. and Veloso, M. (2004), Skill acquisition and use for a dynamically-balancing soccer robot, *in* 'AAAI', pp. 599–604.

Budzianowski, P. and Vulić, I. (2019), 'Hello, it's gpt-2–how can i help you? towards the use of pretrained language models for task-oriented dialogue systems', *arXiv preprint arXiv:1907.05774* .

Burges, C. J. (1998), 'A tutorial on support vector machines for pattern recognition', *Data mining and knowledge discovery* **2**(2), 121–167.

Burman, S., Kim, Y. S., Kang, B. H. and Park, G.-C. (2006), 'Maintenance of game character's ai by players', *International Journal of Multimedia and Ubiquitous Engineering* **1**(1), 39–46.

Calinon, S. and Billard, A. (2007), Incremental learning of gestures by imitation in a humanoid robot, *in* 'Proceedings of the ACM/IEEE international conference on Human-robot interaction', ACM, pp. 255–262.

Calinon, S., Guenter, F. and Billard, A. (2007), 'On learning, representing, and generalizing a task in a humanoid robot', *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* **37**(2), 286–298.

Casadei, R., Fortino, G., Pianini, D., Russo, W., Savaglio, C. and Viroli, M. (2019), 'Modelling and simulation of opportunistic iot services with aggregate computing', *Future Generation Computer Systems* **91**, 252–262.

Cavnar, W. B., Trenkle, J. M. et al. (1994), N-gram-based text categorization, *in* 'Proceedings of SDAIR-94, 3rd annual symposium on document analysis and information retrieval', Vol. 161175, Citeseer.

Charniak, E. (2000), A maximum-entropy-inspired parser, *in* 'Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference', Association for Computational Linguistics, pp. 132–139.

Chaw, S. Y. (2009), Addressing the brittleness of knowledge-based question-answering, PhD thesis.

Chen, L., Gunduz, S. and Ozsu, M. T. (2006), Mixed type audio classification with support vector machine, *in* '2006 IEEE International Conference on Multimedia and Expo', IEEE, pp. 781–784.

Chen, M., Foueughi, E. H. F., Huang, Z. X., Kapetanakis, S., Kostiadis, K., Kummeneje, J., Noda, I., Obst, O., Riley, P., Steffens, T., Wang, Y. and Yin, X. (2018), 'Robocup soccer server users manual'. [Accessed August 23rd, 2018].
URL: *http://sourceforge.net/projects/sserver*

Chen, W., Ananthakrishnan, S., Kumar, R., Prasad, R. and Natarajan, P. (2013), Asr error detection in a conversational spoken language translation system, *in* '2013 IEEE International Conference on Acoustics, Speech and Signal Processing', pp. 7418–7422.

Chernova, S. and Thomaz, A. L. (2014), 'Robot learning from human teachers', *Synthesis Lectures on Artificial Intelligence and Machine Learning* **8**(3), 1–121.

Chung, H., Chen, R., Han, S. C. and Kang, B. H. (2016), Combining rdr-based machine learning approach and human expert knowledge for phishing prediction, *in* 'Pacific Rim International Conference on Artificial Intelligence', Springer, pp. 80–92.

Clark, P., Harrison, P., Jenkins, T., Thompson, J. A., Wojcik, R. H. et al. (2005), Acquiring and using world knowledge using a restricted subset of english., *in* 'FLAIRS Conference', pp. 506–511.

Coalson, J. (2016), 'Free lossless audio codec (flac)'. [Accessed 23 Nov 2016].
URL: *https://xiph.org/flac/format.html*

Colby, K. M. (1971), 'Artificial paranoia', *Artificial intelligence* **2**(1), 1–25.

Compton, P. (2011), Challenges with rules, Technical report, Pacific Knowledge Systems. Retrieved December 12, 2017 from http://pks.com.au/technology/resources/.

Compton, P., Edwards, G., Kang, B., Lazarus, L., Malor, R., Preston, P. and Srinivasan, A. (1992), 'Ripple down rules: Turning knowledge acquisition into knowledge maintenance', *Artificial Intelligence in Medicine* **4**(6), 463–475.

Compton, P. and Jansen, R. (1988), Knowledge in context: A strategy for expert system maintenance, *in* 'Australian Joint Conference on Artificial Intelligence', Springer, pp. 292–306.

Compton, P., Peters, L., Edwards, G. and Lavers, T. G. (2006), 'Experience with ripple-down rules', *Knowledge-Based Systems* **19**(5), 356–362.

Compton, P., Ramadan, Z., Preston, P., Le-Gia, T., Chellen, V. and Mullholland, M. (1998), A trade-off between domain knowledge and problem solving method power, *in* '11th Banff KAW Proceeding', pp. 1–19.

Compton, P. and Richards, D. (1999), Extending ripple down rules, *in* 'Australian Knowledge Acquisition Workshop', Vol. 99, Citeseer.

Compton, P. and Richards, D. (2000), Generalising ripple-down rules, *in* 'International Conference on Knowledge Engineering and Knowledge Management', Springer, pp. 380–386.

Cormen, T. H., Leiserson, C. E., Rivest, R. L. and Stein, C. (2009), *Introduction to algorithms*, MIT press.

Cunningham, H., Maynard, D., Bontcheva, K. and Tablan, V. (2002), A framework and graphical development environment for robust nlp tools and applications., *in* 'ACL', pp. 168–175.

Cunningham, H., Maynard, D. and Tablan, V. (1999), Jape: a java annotation patterns engine, Research memorandum cs-99-06, Department of Computer Science, University of Sheffield.

Dautenhahn, K. and Nehaniv, C. L. (2002), The correspondence problem, *in* 'Imitation in Animals and Artifacts', MIT Press.

Dazeley, R., Warner, P., Johnson, S. and Vamplew, P. (2010), The ballarat incremental knowledge engine, *in* 'Pacific Rim Knowledge Acquisition Workshop', Springer, pp. 195–207.

de Barcelos Silva, A., Gomes, M. M., da Costa, C. A., da Rosa Righi, R., Barbosa, J. L. V., Pessin, G., De Doncker, G. and Federizzi, G. (2020), 'Intelligent personal assistants: A systematic literature review', *Expert Systems with Applications* p. 113193.

De Marneffe, M.-C., MacCartney, B., Manning, C. D. et al. (2006), Generating typed dependency parses from phrase structure parses., *in* 'Lrec', Vol. 6, pp. 449–454.

D'Este, C., Reid, D. and Kang, B. H. (2008), A robotic interface to a medication review expert system, *in* 'Ubiquitous Multimedia Computing, 2008. UMC'08. International Symposium on', IEEE, pp. 145–150.

D'Este, C. and Sammut, C. (2008), 'Learning and generalising semantic knowledge from object scenes', *Robotics and Autonomous Systems* **56**(11), 891–900.

Dhaliwal, J. S. (1996), 'The use and effects of knowledge-based system explanations: Theoretical foundations and a framework for empirical evaluation', *Information systems research* **7**(3), 342–362.

Dice, L. R. (1945), 'Measures of the amount of ecologic association between species', *Ecology* **26**(3), 297–302.

Dickens, C. (1843), 'A christmas carol'. [Accessed 15 Dec 2017].
URL: *https://www.gutenberg.org/ebooks/46*

Dizon, G. (2017), 'Using intelligent personal assistants for second language learning: A case study of alexa', *TESOL Journal* **8**(4), 811–830.

Dizon, G. and Tang, D. (2019), 'A pilot study of alexa for autonomous second language learning', *CALL and complexity–short papers from EUROCALL 2019* p. 107.

Doddington, G. (2002), Automatic evaluation of machine translation quality using n-gram co-occurrence statistics, *in* 'Proceedings of the second international conference on Human Language Technology Research', pp. 138–145.

Dua, D. and Graff, C. (2017), 'UCI machine learning repository'.
URL: *http://archive.ics.uci.edu/ml*

Duda, R., Gaschnig, J. and Hart, P. (1979), 'Model design in the prospector consultant system for mineral exploration', *Expert systems in the microelectronic age* **1234**, 153–167.

Edwards, G., Compton, P., Malor, R., Srinivasan, A. and Lazarus, L. (1993), 'Peirs: a pathologist-maintained expert system for the interpretation of chemical pathology reports', *Pathology* **25**(1), 27–34.

Errattahi, R., Hannani, A. E. and Ouahmane, H. (2018), 'Automatic speech recognition errors detection and correction: A review', *Procedia Computer Science* **128**, 32 – 37. 1st International Conference on Natural Language and Speech Processing.

Esmaili, N., Sammut, C. and Shirazi, G. M. (1995), Behavioural cloning in control of a dynamic system, *in* '1995 IEEE International Conference on Systems, Man and Cybernetics. Intelligent Systems for the 21st Century', Vol. 3, IEEE, pp. 2904–2909.

Fader, A., Zettlemoyer, L. and Etzioni, O. (2013), Paraphrase-driven learning for open question answering, *in* 'Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)', pp. 1608–1618.

Fattah, P., Aickelin, U. and Wagner, C. (2016), 'Optimising rule-based classification in temporal data', *arXiv preprint arXiv:1607.05913* .

Fei, Y., Lei, J., Chengyu, Z., Wu, C., Chengyu, Z. et al. (2011), 'Real-time virtual reference service based on applicable artificial intelligence technologies: The début of the robot xiaotu at tsinghua university library'.

Feng, H., Fawaz, K. and Shin, K. G. (2017), Continuous authentication for voice assistants, *in* 'Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking', ACM, pp. 343–355.

Figueroa, A. (2017), 'Automatically generating effective search queries directly from community question-answering questions for finding related questions', *Expert Systems with Applications* **77**, 11–19.

Finlayson, A. (2008), Incremental knowledge acquisition for complex multi-agent environments., PhD thesis, University of New South Wales, Sydney, Australia.

Fischer, K., Kirstein, F., Jensen, L. C., Krüger, N., Kukliński, K., aus der Wieschen, M. V. and Savarimuthu, T. R. (2016), A comparison of types of robot control for programming by demonstration, *in* '2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI)', IEEE, pp. 213–220.

Fiscus, J. G. (1997), A post-processing system to yield reduced word error rates: Recognizer output voting error reduction (rover), *in* '1997 IEEE Workshop on Automatic Speech Recognition and Understanding Proceedings', IEEE, pp. 347–354.

Fiscus, J. G., Ajot, J., Radde, N. and Laprun, C. (2006), Multiple dimension levenshtein edit distance calculations for evaluating asr systems during simultaneous speech, *in* 'in Proc LREC'.

Fivez, P., Šuster, S. and Daelemans, W. (2017), Unsupervised context-sensitive spelling correction of clinical free-text with word and character n-gram embedding, *in* '16th Workshop on Biomedical Natural Language Processing of the Association for Computational Linguistics', pp. 143–148.

Fonte, F. A. M., Rial, J. C. B., Llamas-Nistal, M. and Hermida, D. F. (2009), Using semantics in ines, an intelligent educational system, *in* '2009 39th IEEE Frontiers in Education Conference', IEEE, pp. 1–6.

Frank, E., Hall, M. and Witten, I. (2016), 'The weka workbench. online appendix for "data mining: Practical machine learning tools and techniques", morgan kaufmann'.

French, J. C., Powell, A. L. and Schulman, E. (1997), Applications of approximate word matching in information retrieval, *in* 'Proceedings of the Sixth International Conference on Information and Knowledge Management', CIKM '97, Association for Computing Machinery, New York, NY, USA, pp. 9–15.

Frey, C. B. and Osborne, M. A. (2017), 'The future of employment: how susceptible are jobs to computerisation?', *Technological Forecasting and Social Change* **114**, 254–280.

Gabrilovich, E. and Markovitch, S. (2006), Overcoming the brittleness bottleneck using wikipedia: Enhancing text categorization with encyclopedic knowledge, *in* 'AAAI', Vol. 6, pp. 1301–1306.

Gadd, T. (1988), "fisching fore werds': phonetic retrieval of written text in information systems', *Program: Automated library and information systems* **22**(3), 222–237.

Gadd, T. (1990), 'Phonix: The algorithm', *Program: Automated library and information systems* **24**(4), 363–366.

Gaines, B. R. (1989), An ounce of knowledge is worth a ton of data: quantitative studies of the trade-off between expertise and data based on statistically well-founded empirical induction, *in* 'Proceedings of the sixth international workshop on Machine learning', Elsevier, pp. 156–159.

Gaines, B. R. and Compton, P. (1995), 'Induction of ripple-down rules applied to modeling large databases', *Journal of Intelligent Information Systems* **5**(3), 211–228.

Gaines, B. R. and Shaw, M. L. G. (1993), 'Knowledge acquisition tools based on personal construct psychology', *The Knowledge Engineering Review* **8**(1), 49–85.

Galgani, F., Compton, P. and Hoffmann, A. (2015), 'LEXA: Building knowledge bases for automatic legal citation classification', *Expert Systems with Applications* **42**(17), 6391 – 6407.

Glina, E. M. and Kang, B. H. (2010), 'Conversation system with state information', *Journal of Advanced Computational Intelligence* **14**(6), 741–745.

Google (2018*a*), 'Google actions sdk'. [Accessed 7 Aug 2018].
URL: *https://developers.google.com/actions/extending-the-assistant*

Google (2018*b*), 'Google speech api'. [Accessed 20 Oct 2018].
URL: *https://w3c.github.io/speech-api/*

Griol, D., Hurtado, L. F., Segarra, E. and Sanchis, E. (2008), 'A statistical approach to spoken dialog systems design and evaluation', *Speech Communication* **50**(8), 666 – 682. Evaluating new methods and models for advanced speech-based interactive systems.
URL: *http://www.sciencedirect.com/science/article/pii/S0167639308000459*

Grosz, B. J., Appelt, D. E., Martin, P. A. and Pereira, F. C. (1987), 'Team: an experiment in the design of transportable natural-language interfaces', *Artificial Intelligence* **32**(2), 173–243.

Gu, X., Herbert, D., Wong, K. C. and Kang, B. (2019), Intelligent web-based task-oriented language assistant using multiple classification ripple down rules, *in* '17th International Conference on Computers, Communications, and Systems', Daegu University, Gyeongsan, South Korea.

Hakak, S. I., Kamsin, A., Shivakumara, P., Gilkar, G. A., Khan, W. Z. and Imran, M. (2019), 'Exact string matching algorithms: Survey, issues, and future research directions', *IEEE Access* **7**, 69614–69637.

Hall, P. A. and Dowling, G. R. (1980), 'Approximate string matching', *ACM computing surveys (CSUR)* **12**(4), 381–402.

Hamade, R. F., Moulianitis, V. C., D'Addonna, D. and Beydoun, G. (2010), 'A dimensional tolerancing knowledge management system using nested ripple down rules (nrdr)', *Engineering Applications of Artificial Intelligence* **23**(7), 1140–1148.

Hamming, R. W. (1950), 'Error detecting and error correcting codes', *The Bell system technical journal* **29**(2), 147–160.

Han, S. C., Mirowski, L., Jeon, S.-H., Lee, G.-S., Kang, B. H. and Turner, P. (2013), Expert systems and home-based telehealth: Exploring a role for mcrdr in enhancing diagnostics, *in* 'International Conference, UCMA, SIA, CCSC, ACIT 2013', Vol. 22, pp. 121–127.

Han, S. C., Mirowski, L. and Kang, B. H. (2015), 'Exploring a role for mcrdr in enhancing telehealth diagnostics', *Multimedia Tools and Applications* **74**(19), 8467–8481.

Hancock, B., Bordes, A., Mazare, P.-E. and Weston, J. (2019), 'Learning from dialogue after deployment: Feed yourself, chatbot!', *arXiv preprint arXiv:1901.05415* .

Hayes, P. J. (1981), The frame problem and related problems in artificial intelligence, *in* 'Readings in artificial intelligence', Elsevier, pp. 223–230.

He, X., Deng, L. and Acero, A. (2011), Why word error rate is not a good metric for speech recognizer training for the speech translation task?, *in* '2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)', IEEE, pp. 5632–5635.

Heckerman, D. (1986), Probabilistic interpretations for mycin's certainty factors, *in* 'Machine Intelligence and Pattern Recognition', Vol. 4, Elsevier, pp. 167–196.

Hendrix, G. G., Sacerdoti, E. D., Sagalowicz, D. and Slocum, J. (1978), 'Developing a natural language interface to complex data', *ACM Transactions on Database Systems (TODS)* **3**(2), 105–147.

Herbert, D. and Kang, B. (2019), Comparative analysis of intelligent personal agent performance, *in* 'Pacific Rim Knowledge Acquisition Workshop', Springer, pp. 127–141.

Herbert, D. and Kang, B. H. (2018), 'Intelligent conversation system using multiple classification ripple down rules and conversational context', *Expert Systems with Applications* **112**, 342 – 352.

Ho Kang, B., Yoshida, K., Motoda, H. and Compton, P. (1997), 'Help desk system with intelligent interface', *Applied Artificial Intelligence* **11**(7-8), 611–631.

Hood, D. (2002), 'Caverphone: Phonetic matching algorithm', *Technical Paper CTP060902, University of Otago, New Zealand* .

Hood, D. (2004), 'Caverphone revisited', *Technical Paper CTP150804* **10**.

Horwitz, J. (2018), 'Siri, alexa, and google assistant can be controlled by inaudible commands'. [15 May, 2018].
URL: *https://venturebeat.com/2018/05/10*

Hoy, M. B. (2018), 'Alexa, siri, cortana, and more: An introduction to voice assistants', *Medical Reference Services Quarterly* **37**(1), 81–88. PMID: 29327988.
URL: *https://doi.org/10.1080/02763869.2018.1404391*

Huang, S., Wu, Y., Wei, F. and Luan, Z. (2019), Dictionary-guided editing networks for paraphrase generation, *in* 'Proceedings of the AAAI Conference on Artificial Intelligence', Vol. 33, pp. 6546–6553.

Hunt, E., Janamsetty, R., Kinares, C., Koh, C., Sanchez, A., Zhan, F., Ozdemir, M., Waseem, S., Yolcu, O., Dahal, B. et al. (2019), Machine learning models for paraphrase identification and its applications on plagiarism detection, *in* '2019 IEEE International Conference on Big Knowledge (ICBK)', IEEE, pp. 97–104.

Hussain, S., Sianaki, O. A. and Ababneh, N. (2019), A survey on conversational agents/chatbots classification and design techniques, *in* 'Workshops of the International Conference on Advanced Information Networking and Applications', Springer, pp. 946–956.

Hyeon, J., Oh, K.-J., Kim, Y. J., Chung, H., Kang, B. H. and Choi, H.-J. (2016), Constructing an initial knowledge base for medical domain expert system using induct rdr, *in* '2016 International Conference on Big Data and Smart Computing (BigComp)', IEEE, pp. 408–410.

IBM (2020), 'Ibm ilog cplex optimization studio'. [Accessed 19 June 2020].
URL: *https://www.ibm.com/au-en/products/ilog-cplex-optimization-studio*

Ikegami, Y., Knauf, R., Damiani, E., Tsuruta, S., Sakurai, Y., Sakurai, E., Kutics, A. and Nakagawa, A. (2019), High performance personal adaptation speech recognition framework by incremental learning with plural language models, *in* '2019 15th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS)', IEEE, pp. 470–474.

Järvelin, A., Järvelin, A. and Järvelin, K. (2007), 's-grams: Defining generalized n-grams for information retrieval', *Information Processing & Management* **43**(4), 1005–1019.

Järvelin, A., Keskustalo, H., Sormunen, E., Saastamoinen, M. and Kettunen, K. (2016), 'Information retrieval from historical newspaper collections in highly inflectional languages: A query expansion approach', *Journal of the Association for Information Science and Technology* **67**(12), 2928–2946.

Jeanrenaud, P., Eide, E., Chaudhari, U., McDonough, J., Ng, K., Siu, M. and Gish, H. (1995), Reducing word error rate on conversational speech from the switchboard corpus, *in* '1995 International Conference on Acoustics, Speech, and Signal Processing', Vol. 1, IEEE, pp. 53–56.

Jiang, J., Hassan Awadallah, A., Jones, R., Ozertem, U., Zitouni, I., Gurunath Kulkarni, R. and Khan, O. Z. (2015), Automatic online evaluation of intelligent assistants, *in* 'Proceedings of the 24th International Conference on World Wide Web', International World Wide Web Conferences Steering Committee, pp. 506–516.

Johannink, T., Bahl, S., Nair, A., Luo, J., Kumar, A., Loskyll, M., Ojea, J. A., Solowjow, E. and Levine, S. (2019), Residual reinforcement learning for robot control, *in* '2019 International Conference on Robotics and Automation (ICRA)', pp. 6023–6029.

Kang, B., Compton, P. and Preston, P. (1995), Multiple classification ripple down rules: evaluation and possibilities, *in* 'Proceedings 9th Banff knowledge acquisition for knowledge based systems workshop', pp. 17.1–17.20.

Kang, B. H. (1995), Validating knowledge acquisition: Multiple classification ripple down rules, PhD thesis, University of New South Wales.

Kang, B. H., Gambetta, W. and Compton, P. (1996), 'Verification and validation with ripple-down rules', *International Journal of Human-Computer Studies* **44**(2), 257–269.

Kasilingam, D. L. (2020), 'Understanding the attitude and intention to use smartphone chatbots for shopping', *Technology in Society* **62**, 101280.

Kate, R. J., Wong, Y. W. and Mooney, R. J. (2005), Learning to transform natural to formal languages, *in* 'Proceedings of the National Conference on Artificial Intelligence', Vol. 20, Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, p. 1062.

Këpuska, V. and Bohouta, G. (2017), 'Comparing speech recognition systems (microsoft api, google api and cmu sphinx)', *Int. J. Eng. Res. Appl* **7**, 20–24.

Keskustalo, H., Pirkola, A., Visala, K., Leppänen, E. and Järvelin, K. (2003), Non-adjacent digrams improve matching of cross-lingual spelling variants, *in* 'International symposium on string processing and information retrieval', Springer, pp. 252–265.

Khan, A., Baharudin, B., Lee, L. H. and Khan, K. (2010), 'A review of machine learning algorithms for text-documents classification', *Journal of advances in information technology* **1**(1), 4–20.

Khan, M., Goskula, T., Nasiruddin, M. and Quazi, R. (2011), 'Comparison between k-nn and svm method for speech emotion recognition', *International Journal on Computer Science and Engineering* **3**(2), 607–611.

Kilgarriff, A. (2006), 'Bnc database and word frequency lists'. [Accessed 1 Feb 2018].
URL: *http://www.kilgarriff.co.uk/bnc-readme.html*

Kim, D., Han, S. C., Lin, Y., Kang, B. H. and Lee, S. (2018), 'Rdr-based knowledge based system to the failure detection in industrial cyber physical systems', *Knowledge-Based Systems* **150**, 1–13.

Kim, Y. J., Hyeon, J., Oh, K. J., Choi, H. J. et al. (2016), Medical prognosis generation in blood total test results, *in* 'The 29th anniversary of the Australasian Joint Conference on Artificial Intelligence', School of Engineering and ICT University of Tasmania.

Klopfenstein, L. C., Delpriori, S., Malatini, S. and Bogliolo, A. (2017), The rise of bots: A survey of conversational interfaces, patterns, and paradigms, *in* 'Proceedings of the 2017 conference on designing interactive systems', pp. 555–565.

Kober, J., Bagnell, J. A. and Peters, J. (2013), 'Reinforcement learning in robotics: A survey', *The International Journal of Robotics Research* **32**(11), 1238–1274.

Kober, J. and Peters, J. (2009), Learning motor primitives for robotics, *in* '2009 IEEE International Conference on Robotics and Automation', pp. 2112–2118.

Kollar, T., Tellex, S., Roy, D. and Roy, N. (2010), Toward understanding natural language directions, *in* '2010 5th ACM/IEEE International Conference on Human-Robot Interaction (HRI)', IEEE, pp. 259–266.

Kondrak, G. (2005), N-gram similarity and distance, *in* 'International symposium on string processing and information retrieval', Springer, pp. 115–126.

Koneru, K., Pulla, V. S. V. and Varol, C. (2016), Performance evaluation of phonetic matching algorithms on english words and street names, *in* 'Proceedings of the 5th International Conference on Data Management Technologies and Applications', SCITEPRESS-Science and Technology Publications, Lda, pp. 57–64.

Kruskal, J. B. (1983), 'An overview of sequence comparison: Time warps, string edits, and macromolecules', *SIAM review* **25**(2), 201–237.

Kubrick, S. (1968), '2001: A space odyssey'. Film, Metro-Goldwyn-Mayer, USA.

Kuhlmann, G., Stone, P., Mooney, R. and Shavlik, J. (2004), Guiding a reinforcement learner with natural language advice: Initial results in robocup soccer, *in* 'The AAAI-2004 workshop on supervisory control of learning and adaptive systems', San Jose, CA.

Kwok, R. B. (2002), Using ripple down rules for actions and planning, *in* 'Pacific Rim International Conference on Artificial Intelligence', Springer, pp. 604–604.

Ledley, R. S. and Lusted, L. B. (1959), 'Reasoning foundations of medical diagnosis', *Science* **130**(3366), 9–21.

Lenat, D. B., Prakash, M. and Shepherd, M. (1985), 'Cyc: Using common sense knowledge to overcome brittleness and knowledge acquisition bottlenecks', *AI magazine* **6**(4), 65–65.

Levenshtein, V. I. (1966), Binary codes capable of correcting deletions, insertions, and reversals, *in* 'Soviet physics doklady', Vol. 10, pp. 707–710.

Li, B., Sainath, T., Narayanan, A., Caroselli, J., Bacchiani, M., Misra, A., Shafran, I., Sak, H., Pundak, G., Chin, K. et al. (2017), 'Acoustic modeling for google home', *INTERSPEECH-2017* pp. 399–403.

Li, F. and Jagadish, H. (2014*a*), 'Constructing an interactive natural language interface for relational databases', *Proceedings of the VLDB Endowment* **8**(1), 73–84.

Li, F. and Jagadish, H. V. (2014*b*), Nalir: an interactive natural language interface for querying relational databases, *in* 'Proceedings of the 2014 ACM SIGMOD international conference on Management of data', ACM, pp. 709–712.

Li, Y., Yang, H. and Jagadish, H. (2005), Nalix: an interactive natural language interface for querying xml, *in* 'Proceedings of the 2005 ACM SIGMOD international conference on Management of data', ACM, pp. 900–902.

Liao, S.-H. (2005), 'Expert system methodologies and applications—a decade review from 1995 to 2004', *Expert systems with applications* **28**(1), 93–103.

Lindsay, R. K., Buchanan, B. G., Feigenbaum, E. A. and Lederberg, J. (1980), 'Applications of artificial intelligence for organic chemistry: the dendral project', *New York* .

Loftin, R. T., MacGlashan, J., Peng, B., Taylor, M. E., Littman, M. L., Huang, J. and Roberts, D. L. (2014), A strategy-aware technique for learning behaviors from discrete human feedback, *in* 'Twenty-Eighth AAAI Conference on Artificial Intelligence'.

Lopatovska, I., Rink, K., Knight, I., Raines, K., Cosenza, K., Williams, H., Sorsche, P., Hirsch, D., Li, Q. and Martinez, A. (2018), 'Talk to me: Exploring user interactions with the amazon alexa', *Journal of Librarianship and Information Science* .
URL: *https://doi.org/10.1177/0961000618759414*

Lucas, G. (1977), 'Star wars: Episode iv – a new hope'. Film, 20th Century Fox, USA.

Madnani, N. and Dorr, B. J. (2010), 'Generating phrasal and sentential paraphrases: A survey of data-driven methods', *Computational Linguistics* **36**(3), 341–387.

Mahapatra, R., Sharma, N., Trivedi, A. and Aman, C. (2012), Adding interactive interface to e-government systems using aiml based chatterbots, *in* '2012 CSI Sixth International Conference on Software Engineering (CONSEG)', IEEE, pp. 1–6.

Majid al Rifaie, M. (2018), 'Loebner prize in artificial intelligence'. [Accessed 27 Sep 2018],Society for the Study of Artificial Intelligence and the Simulation of Behaviour (AISB).
URL: *https://www.aisb.org.uk/events/loebner-prize*

Mak, P., Kang, B.-H., Sammut, C. and Kadous, W. (2004), Knowledge acquisition module for conversation agent, Technical report, School of Computing, University of Tasmania.

Mangu, L. and Padmanabhan, M. (2001), Error corrective mechanisms for speech recognition, *in* '2001 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No.01CH37221)', Vol. 1, pp. 29–32 vol.1.

Manikonda, L., Deotale, A. and Kambhampati, S. (2017), 'What's up with privacy?: User preferences and privacy concerns in intelligent personal assistants', *arXiv preprint arXiv:1711.07543* .

Manning, C., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S. and McClosky, D. (2014), The stanford corenlp natural language processing toolkit, *in* 'Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations', pp. 55–60.

Marcus, G. (2019), 'Deep understanding: The next challenge for ai', *Proceedings from NeurIPS* .

Marietto, M. d. G. B., de Aguiar, R. V., Barbosa, G. d. O., Botelho, W. T., Pimentel, E., França, R. d. S. and da Silva, V. L. (2013), 'Artificial intelligence markup language: a brief tutorial', *arXiv preprint arXiv:1307.3091* .

Matuszek, C. (2015), Talking to Robots: Learning to Ground Human Language in Perception and Execution, Thesis, Computer Science and Engineering, University of Washington.

Matuszek, C., Herbst, E., Zettlemoyer, L. and Fox, D. (2013), Learning to parse natural language commands to a robot control system, *in* 'Experimental Robotics', Springer, pp. 403–415.

Mauldin, M. L. (1994), Chatterbots, tinymuds, and the turing test: Entering the loebner prize competition, *in* 'AAAI', Vol. 94, pp. 16–21.

McCarthy, J., Minsky, M. L., Rochester, N. and Shannon, C. E. (2006), 'A proposal for the dartmouth summer research project on artificial intelligence, august 31, 1955', *AI magazine* **27**(4), 12.

Medscope (2020), 'Medication review mentor'. [Accessed 5 May 2020].
URL: *https://www.medscope.com.au*

Mehr, H., Ash, H. and Fellow, D. (2017), 'Artificial intelligence for citizen services and government', *Ash Cent. Democr. Gov. Innov. Harvard Kennedy Sch., no. August* pp. 1–12.

Microsoft (2018), 'Cortana'. [Accessed 18 Sep 2018].
URL: *https://developer.microsoft.com/en-us/cortana*

Mikic, F. A., Burguillo, J. C., Llamas, M., Rodríguez, D. A. and Rodríguez, E. (2009), Charlie: An aiml-based chatterbot which works as an interface among ines and humans, *in* '2009 EAEEIE Annual Conference', IEEE, pp. 1–6.

Mikic, F. A., Burguillo, J. C., Rodríguez, D. A., Rodríguez, E. and Llamas, M. (2008), T-bot and q-bot: A couple of aiml-based bots for tutoring courses and evaluating students, *in* '2008 38th Annual Frontiers in Education Conference', IEEE, pp. S3A–7.

Miller, G. A. (1995), 'Wordnet: a lexical database for english', *Communications of the ACM* **38**(11), 39–41.

Minksy, M. (1975), 'A framework for representing knowledge', *The psychology of computer vision* **73**, 211–277.

Miranda-Mena, T. G., Ochoa, J. L., Martínez-Béjar, R., Fernández-Breis, J. T. and Salinas, J. (2006), 'A knowledge-based approach to assign breast cancer treatments in oncology units', *Expert systems with applications* **31**(3), 451–457.

Miranda-Mena, T. G., U., S. L. B., Ochoa, J. L., Martínez-Béjar, R., Fernández-Breis, J. T. and Salinas, J. (2006), 'A knowledge-based approach to assign breast cancer treatments in oncology units', *Expert Systems with Applications* **31**(3), 451 – 457.

Mnasri, M. (2019), 'Recent advances in conversational nlp: Towards the standardization of chatbot building', *arXiv preprint arXiv:1903.09025* .

Mokotoff, G. (1997), 'Soundex genealogy'. [Accessed 5 May 2020].
URL: *https://www.avotaynu.com/soundex.htm*

Montenegro, J. L. Z., da Costa, C. A. and da Rosa Righi, R. (2019), 'Survey of conversational agents in health', *Expert Systems with Applications* .

Moore, A., Parada, P. P. and Naylor, P. (2017), 'Speech enhancement for robust automatic speech recognition: Evaluation using a baseline system and instrumental measures', *Computer Speech and Language* **46**, 574 – 584.

Mulholland, M., McKinnon, K. and Haddad, P. (1996), 'Practical evaluation of ion chromatography methods developed by an expert system', *Journal of Chromatography A* **739**(1-2), 25–33.

Mulholland, M., Preston, P., Sammut, C., Hibbert, B. and Compton, P. (1993), An expert system for ion chromatography developed using machine learning and knowledge in context, *in* 'Proceedings of the 6th international conference on Industrial and engineering applications of artificial intelligence and expert systems', pp. 258–267.

Munoz-Morera, J., Maza, I., Fernandez-Aguera, C. J., Caballero, F. and Ollero, A. (2015), Assembly planning for the construction of structures with multiple uas equipped with robotic arms, *in* 'Unmanned Aircraft Systems (ICUAS), 2015 International Conference on', IEEE, pp. 1049–1058.

Natcorp (2018), 'British national corpus [bnc]'. [Accessed 15 Dec 2017].
URL: *http://www.natcorp.ox.ac.uk*

Navarro, G. (2001), 'A guided tour to approximate string matching', *ACM computing surveys (CSUR)* **33**(1), 31–88.

Needleman, S. B. and Wunsch, C. D. (1970), 'A general method applicable to the search for similarities in the amino acid sequence of two proteins', *Journal of molecular biology* **48**(3), 443–453.

Nehmzow, U., Akanyeti, O., Weinrich, C., Kyriacou, T. and Billings, S. A. (2007), Robot programming by demonstration through system identification, *in* 'Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on', IEEE, pp. 801–806.

Newell, A. and Simon, H. A. (1976), 'Computer science as empirical inquiry: Symbols and search', *Communications of the ACM* **19**(3), 113–126.

Nguyen, D. Q., Nguyen, D. Q., Pham, D. D. and Pham, S. B. (2016), 'A robust transformation-based learning approach using ripple down rules for part-of-speech tagging', *AI communications* **29**(3), 409–422.

Nguyen, D. Q., Nguyen, D. Q. and Pham, S. B. (2017), 'Ripple down rules for question answering', *Semantic Web* **8**(4), 511–532.

Oh, K.-J., Choi, H.-J., Gweon, G., Heo, J. and Ryu, P.-M. (2015), Paraphrase generation based on lexical knowledge and features for a natural language question answering system, *in* 'Big Data and Smart Computing (BigComp), 2015 International Conference on', IEEE, pp. 35–38.

Oh, Y. H., Chung, K., Ju, D. Y. et al. (2020), 'Differences in interactions with a conversational agent', *International Journal of Environmental Research and Public Health* **17**(9), 3189.

OpenCyc (2018). [Accessed 3 June 2020].
URL: *https://github.com/asanchez75/opencyc*

O'Shaughnessy, D. (2008), 'Invited paper: Automatic speech recognition: History, methods and challenges', *Pattern Recognition* **41**(10), 2965 – 2979.

O'Shea, K. (2014), 'Natural language scripting within conversational agent design', *Applied Intelligence* **40**(1), 189–197.
URL: *http://dx.doi.org/10.1007/s10489-012-0408-2*

Pal, D., Arpnikanondt, C., Funilkul, S. and Varadarajan, V. (2019), User experience with smart voice assistants: The accent perspective, *in* '2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT)', IEEE, pp. 1–6.

Papineni, K., Roukos, S., Ward, T. and Zhu, W.-J. (2002), Bleu: a method for automatic evaluation of machine translation, *in* 'Proceedings of the 40th annual meeting on association for computational linguistics', Association for Computational Linguistics, pp. 311–318.

Park, Y., Patwardhan, S., Visweswariah, K. and Gates, S. C. (2008), An empirical analysis of word error rate and keyword error rate, *in* 'Ninth Annual Conference of the International Speech Communication Association'.

Pelk, H. (2016), 'Chatbots: A bright future in iot?'. [Accessed 19 June 2020].
URL: *https://itnext.io/chatbots-a-bright-future-in-iot-f9f6cc8e12a9*

Pellegrini, T. and Trancoso, I. (2010), Improving asr error detection with non-decoder based features, *in* 'Eleventh Annual Conference of the International Speech Communication Association', pp. 1950–1953.

Peng, B., Zhu, C., Li, C., Li, X., Li, J., Zeng, M. and Gao, J. (2020), 'Few-shot natural language generation for task-oriented dialog', *arXiv preprint arXiv:2002.12328* .

Pfeifer, U., Poersch, T. and Fuhr, N. (1996), 'Retrieval effectiveness of proper name search methods', *Information Processing & Management* **32**(6), 667–679.

Pham, K. C. and Sammut, C. (2005), Rdrvision-learning vision recognition with ripple down rules, *in* 'Proceedings of the Australasian Conference on Robotics and Automation', p. 7.

Philips, L. (1990), 'Hanging on the metaphone', *Computer Language* **7**(12), 39–43.

Philips, L. (2000*a*), 'The double metaphone search algorithm', *C/C++ users journal* **18**(6), 38–43.

Philips, L. (2000*b*), 'The double metaphone search algorithm'. [Accessed 21 May 2020].
URL: *hhttps://www.drdobbs.com/the-double-metaphone-search-algorithm*

Pierris, G. and Dahl, T. (2017), 'Learning robot control using a hierarchical som-based encoding', *IEEE Transactions on Cognitive and Developmental Systems* **PP**(99), 1–1.

PKS (2020), 'Pacific knowledge systems'. [Accessed 28 May 2020].
URL: *https://www.pks.com.au*

Polit, S. (1984), 'R1 and beyond: Ai technology transfer at digital equipment corporation', *AI Magazine* **5**(4), 76.

Pollack, J. B. (2006), 'Mindless intelligence', *IEEE Intelligent Systems* **21**(3), 50–56.

Popescu, A.-M., Armanasu, A., Etzioni, O., Ko, D. and Yates, A. (2004), Modern natural language interfaces to databases: Composing statistical parsing with semantic tractability, *in* 'Proceedings of the 20th international conference on Computational Linguistics', Association for Computational Linguistics, p. 141.

Preece, A. (2001), Evaluating verification and validation methods in knowledge engineering, *in* 'Industrial Knowledge Management', Springer, pp. 91–104.

Preston, P., Edwards, G. and Compton, P. (1994), A 2000 rule expert system without a knowledge engineer, *in* 'proceedings of the 8th AAAI-sponsored Banff knowledge acquisition for knowledge-based systems workshop, Banff, Canada', Citeseer, pp. 17–1.

Pringle, D. and Pratchett, T. (1998), *The Ultimate Encyclopedia of Fantasy*, Overlook Press.

Protalinski, E. (2017), 'Google's speech recognition technology now has a 4.9% word error rate'. [Accessed 1 February 2018].
URL: *https://venturebeat.com/2017/05/17*

Quinlan, J. R. (1993), *C4.5 : programs for machine learning*, Morgan Kaufmann series in machine learning, Morgan Kaufmann Publishers.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D. and Sutskever, I. (2019), 'Language models are unsupervised multitask learners', *OpenAI Blog* **1**(8), 9.

Raghavan, H. and Allan, J. (2005), Matching inconsistently spelled names in automatic speech recognizer output for information retrieval, *in* 'Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing', Association for Computational Linguistics, pp. 451–458.

Rainbolt, D. (2007), *Ghost Cats : Human Encounters with Feline Spirits*, Stupid Gravity Press.

Reis, A., Paulino, D., Paredes, H. and Barroso, J. (2017), Using intelligent personal assistants to strengthen the elderlies' social bonds, *in* 'International Conference on Universal Access in Human-Computer Interaction', Springer, pp. 593–602.

Revesz, P. and Triplet, T. (2011), 'Temporal data classification using linear classifiers', *Information Systems* **36**(1), 30–41.

Richards, D. (2009), 'Two decades of ripple down rules research', *The Knowledge Engineering Review* **24**(2), 159–184.

Richards, L. E., Nguyen, A. T., Darvish, K., Raff, E. and Matuszek, C. (2019), A manifold alignment approach to grounded language learning, *in* '8th Northeast Robotics Colloquium'.

Ringger, E. K. and Allen, J. F. (1996), Error correction via a post-processor for continuous speech recognition, *in* '1996 IEEE International Conference on Acoustics, Speech, and Signal Processing Conference Proceedings', Vol. 1, pp. 427–430 vol. 1.

Rish, I. et al. (2001), An empirical study of the naive bayes classifier, *in* 'IJCAI 2001 workshop on empirical methods in artificial intelligence', Vol. 3, pp. 41–46.

Robotis (2018), 'Robot operating system (ros)'. [Accessed July 2nd, 2018].
URL: *http://www.ros.org/about-ros/*

Sammut, C., Hurst, S., Kedzier, D. and Michie, D. (1992), Learning to fly, *in* 'Proceedings of the ninth international workshop on Machine learning 1992', pp. 385–393.

Saritas, M. M. and Yasar, A. (2019), 'Performance analysis of ann and naive bayes classification algorithm for data classification', *International Journal of Intelligent Systems and Applications in Engineering* **7**(2), 88–91.

Sarma, A. and Palmer, D. D. (2004), Context-based speech recognition error detection and correction, *in* 'Proceedings of HLT-NAACL 2004: Short Papers', Association for Computational Linguistics, pp. 85–88.

Sarraf, Q. and Ellis, G. (2006), 'Business rules in retail: The tesco.com story', *Business Rules Journal* **7**(6). [Accessed January 3rd, 2017].
URL: *http://www.brcommunity.com/p-n014.php*

SAS (2020), 'Sas business rules manager'. [Accessed 19 June 2020].
URL: *https://www.sas.com/en_au/software/business-rules-manager.html*

Saunders, W., Sastry, G., Stuhlmueller, A. and Evans, O. (2018), Trial without error: Towards safe reinforcement learning via human intervention, *in* 'Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems', International Foundation for Autonomous Agents and Multiagent Systems, pp. 2067–2069.

Scheffer, T. (1996), Algebraic foundation and improved methods of induction of ripple down rules, *in* 'In'.

Sebastiani, F. (2002), 'Machine learning in automated text categorization', *ACM computing surveys (CSUR)* **34**(1), 1–47.

Segway (2018), 'Personal transportation that simply moves you'. [Accessed July 3rd, 2018].
URL: *http://www.segway.com/*

Shabaz, K., O'Shea, J. D., Crockett, K. A. and Latham, A. (2015), Aneesah: A conversational natural language interface to databases, *in* 'World Congress on Engineering', pp. 227–232.

Shah, H., Warwick, K., Vallverdú, J. and Wu, D. (2016), 'Can machines talk? comparison of eliza with modern dialogue systems', *Computers in Human Behavior* **58**, 278 – 295.
URL: *http://www.sciencedirect.com/science/article/pii/S0747563216300048*

Shannon, C. E. (1948), 'A mathematical theory of communication', *Bell system technical journal* **27**(3), 379–423.

Sheh, R., Kadous, M. W., Sammut, C. and Hengst, B. (2007), Extracting terrain features from range images for autonomous random stepfield traversal, *in* 'Safety, Security and Rescue Robotics, 2007. SSRR 2007. IEEE International Workshop on', IEEE, pp. 1–6.

Shiraz, G. and Sammut, C. (1997), Combining knowledge acquisition and machine learning to control dynamic systems, *in* 'International Joint Conference on Artificial Intelligence', Morgan Kaufmann, pp. 908–913.

Shirazi, H. and Sammut, C. (2008), 'Acquiring control knowledge from examples using ripple-down rules and machine learning', *Iranian Journal of Science and Technology* **32**(B3), 295–304.

Shortliffe, E. H. (1973), 'An artificial intelligence program to advise physicians regarding antimicrobial therapy', *Computers and biomedical research* **6**(6), 544–560.

Shortliffe, E. H., Davis, R., Axline, S. G., Buchanan, B. G., Green, C. C. and Cohen, S. N. (1975), 'Computer-based consultations in clinical therapeutics: explanation and rule acquisition capabilities of the mycin system', *Computers and biomedical research* **8**(4), 303–320.

Sidorov, G., Velasquez, F., Stamatatos, E., Gelbukh, A. and Chanona-Hernández, L. (2014), 'Syntactic n-grams as machine learning features for natural language processing', *Expert Systems with Applications* **41**(3), 853–860.

Singer-Vine, J. (2014), 'Markovify'. [Accessed 15 Dec 2017].
URL: *https://github.com/jsvine/markovify*

Smith, E., Crockett, K., Latham, A. and Buckingham, F. (2014), Seeker: A conversational agent as a natural language interface to a relational database, *in* 'Proceedings of the World Congress on Engineering', Newswood/International Association of Engineers, pp. 191–196.

Snover, M., Dorr, B., Schwartz, R., Micciulla, L. and Makhoul, J. (2006), A study of translation edit rate with targeted human annotation, *in* 'Proceedings of association for machine translation in the Americas', Vol. 200.

Sonetto Enhances Online Experience @ tesco.com (2005). [Accessed 28 May 2020].
URL: *https://www.talkingretail.com/news/industry-news/sonetto-enhances-online-experience-tesco-com-24-05-2005/*

Sonos Wireless Speakers (2020). [Accessed 8 June 2020].
URL: *https://www.sonos.com/en-au/home*

Sorensen, T. A. (1948), 'A method of establishing groups of equal amplitude in plant sociology based on similarity of species content and its application to analyses of the vegetation on danish commons', *Biol. Skar.* **5**, 1–34.

Strayer, D. L., Cooper, J. M., Turrill, J., Coleman, J. R. and Hopman, R. J. (2017), 'The smartphone and the driver's cognitive workload: A comparison of apple, google, and microsoft's intelligent personal assistants.', *Canadian Journal of Experimental Psychology/Revue canadienne de psychologie expérimentale* **71**(2), 93.

Suddrey, G., Eich, M., Maire, F. and Roberts, J. (2016), Learning functional argument mappings for hierarchical tasks from situation specific explanations, *in* 'Australasian Joint Conference on Artificial Intelligence', Springer, pp. 345–352.

Suddrey, G., Lehnert, C., Eich, M., Maire, F. and Roberts, J. (2017), 'Teaching robots generalizable hierarchical tasks through natural language instruction', *IEEE Robotics and Automation Letters* **2**(1), 201–208.

Sundman, J. (2003), 'Artificial stupidity'. [Accessed 27 Sep 2018].
URL: *https://www.salon.com/2003/02/26/loebner_part_one/*

Suryanto, H., Richards, D. and Compton, P. (1999), The automatic compression of multiple classification ripple down rule knowledge based systems: preliminary experiments, *in* 'Knowledge-Based Intelligent Information Engineering Systems, Third International Conference', IEEE, pp. 203–206.

Sutskever, I., Vinyals, O. and Le, Q. V. (2014), Sequence to sequence learning with neural networks, *in* 'Advances in neural information processing systems', pp. 3104–3112.

Tellex, S. A., Kollar, T. F., Dickerson, S. R., Walter, M. R., Banerjee, A., Teller, S. and Roy, N. (2011), Understanding natural language commands for robotic navigation and mobile manipulation, *in* 'Proceedings of the National Conference on Artificial Intelligence (AAAI 2011)', pp. pp. 1507–1514.

Thakker, D., Osman, T. and Lakin, P. (2009), 'Gate jape grammar tutorial', *Nottingham Trent University, UK, Phil Lakin, UK, Version* **1**.

The Cyc Platform (2020). [Accessed 3 June 2020].
URL: *https://www.cyc.com/the-cyc-platform*

Tissot, H. and Dobson, R. (2019), 'Combining string and phonetic similarity matching to identify misspelt names of drugs in medical records written in portuguese', *Journal of biomedical semantics* **10**(1), 17.

Torralba-Rodríguez, F. J., Bixquert-Montagud, V., Fernández-Breis, J. T. and Martínez-Béjar, R. (2010), 'An incremental knowledge acquisition-based system for supporting decisions in biomedical domains', *Computer methods and programs in biomedicine* **98**(2), 161–171.

Torrey, L. and Taylor, M. (2013), Teaching on a budget: Agents advising agents in reinforcement learning, *in* 'Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems', pp. 1053–1060.

Turing, A. M. (1950), 'Computing machinery and intelligence', *Mind* **59**(236), 433–460.

Turtlebot3 (2020). [Accessed 10 June 2020].
URL: *https://emanual.robotis.com/docs/en/platform/turtlebot3/overview/*

Ukkonen, E. (1992), 'Approximate string-matching with q-grams and maximal matches', *Theoretical computer science* **92**(1), 191–211.

Vapnik, V. and Vapnik, C. (1995), 'The nature of statistical learning theory support-vector networks', *Mach. Learn* .

Vazey, M. and Richards, D. (2005), Troubleshooting at the call centre: A knowledge-based approach, *in* 'Artificial Intelligence and Applications', pp. 721–726.

Wallace, R. (2011), 'Artificial intelligence markup language (aiml)'. [Accessed 24 Oct 2016].
URL: *http://www.alicebot.org/TR/2011/*

Wallace, R. (2016), 'Alice'. [Accessed 24 Oct 2016].
URL: *http://www.alicebot.org/about.html*

Waltz, D. L. (1978), 'An english language question answering system for a large relational database', *Communications of the ACM* **21**(7), 526–539.

Warren, D. H. and Pereira, F. C. (1982), 'An efficient easily adaptable system for interpreting natural language queries', *Computational Linguistics* **8**(3-4), 110–122.

Wei, Y., Zhu, X., Sun, B. and Sun, B. (2016), Comparative studies of aiml, *in* '2016 3rd International Conference on Systems and Informatics (ICSAI)', IEEE, pp. 344–349.

Weizenbaum, J. (1966), 'Eliza—a computer program for the study of natural language communication between man and machine', *Communications of the ACM* **9**(1), 36–45.

Westerman, D., Cross, A. C. and Lindmark, P. G. (2019), 'I believe in a thing called bot: Perceptions of the humanness of "chatbots"', *Communication Studies* **70**(3), 295–312.

Wilcox, B. (2011), 'Pattern matching for natural language applications'. [Accessed 26 Oct 2016].
URL: *http://chatscript.sourceforge.net/Documentation*

Wille, R. (1992), 'Concept lattices and conceptual knowledge systems', *Computers and mathematics with applications* **23**(6-9), 493–515.

Woods, W. A. and Bobrow, D. G. (1970), 'Transition network grammars for natural language analysis', *Communications of the ACM* **13**(10), 591–606.

Woods, W. A., Kaplan, R. M. and Nash-Webber, B. (1972), *The Lunar Sciences: Natural Language Information System: Final Report*, Bolt Beranek and Newman.

Worswick, S. (2019), 'Mitsuku chatbot'. [Accessed 1 Jun 2020].
URL: *http://www.mitsuku.com*

Yang, W., Herbert, D., Kim, S. and Kang, B. (2019), 'Mcrdr knowledge-based 3d dialogue simulation in clinical training and assessment', *Journal of medical systems* **43**(7), 200.

Yu, D. and Deng, L. (2016), *Automatic Speech Recognition*, Springer.

Zámečníková, E., Kreslíková, J. et al. (2016), 'Business rules definition for decision support system using matrix grammar', *Acta Informatica Pragensia* **5**(1), 72–81.

Zelle, J. M. and Mooney, R. J. (1996), Learning to parse database queries using inductive logic programming, *in* 'Proceedings of the national conference on artificial intelligence', pp. 1050–1055.

Zemalache, K. M. and Maaref, H. (2009), 'Controlling a drone: Comparison between a based model method and a fuzzy inference system', *Applied Soft Computing* **9**(2), 553–562.

Zhou, L., Shi, Y., Feng, J. and Sears, A. (2005), 'Data mining for detecting errors in dictation speech recognition', *IEEE Transactions on Speech and Audio Processing* **13**(5), 681–688.

Zipf, G. K. (1949), *Human behavior and the principle of least effort: An introduction to human ecology*, Addison-Wesley.

Zobel, J. and Dart, P. (1995), 'Finding approximate matches in large lexicons', *Software: Practice and Experience* **25**(3), 331–345.

Zobel, J. and Dart, P. (1996), Phonetic string matching: Lessons from information retrieval, *in* 'Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval', pp. 166–172.

Appendix A

Evaluation Data

A.1 Speech Corrections

Note speech-to-text and text-to-speech corrections in the C-MCRDR CA system were performed on web browser speech API inputs and outputs (Google's Web Speech API (Google, 2018*b*)).

Table A.1: C-MCRDR CA system speech-to-text corrections

ASR Result	Correction
kit (\d)	kit$1
hit (\d)	kit$1
item to	item 2
k na (\d)	kne$1
kna (\d)	kne$1
ke(\d)	kne$1
item for	item 4
engineering (\d)	kne$1
ict (\d)	kit$1
kiit	kit
.*[i][t] (\d)	kit$1
.*[i][k] (\d)	kit$1
[k][ ][i][ ][t] (\d)	kit$1
canny (\d)	kne$1
kenny (\d)	kne$1
kids to (\d)	kit2$1
kids (\d)	kit$1
assignment to	assignment 2
assignment for	assignment 4
assignment (\d) do	assignment $1 due
assignment to do	assignment 2 due
assignment for do	assignment 4 due
assignment (\d) you	assignment $1 due
kicks (\d)	kit$1
kitz (\d)	kit$1

Continued on next page

Table A.1 – *Continued from previous page*

ASR Result	Correction
kiss (\d)	kit$1
kid (\d)	kit$1
keeps (\d)	kit$1
keep (\d)	kit$1
tip (\d)	kit$1
tips (\d)	kit$1
kick (\d)	kit$1
tips to A3	kit203
kit for 18	kit418
Kitty (\d)	kit$1
one	1
two	2
three	3
four	4
five	5
six	6
seven	7
eight	8
nine	9
zero	0
oh	0
k. i. t.	KIT
KIT (\d) (\d) (\d)	KIT$1$2$3
how about	hobart
(\d)(\d)(\d)(\d) (\d)(\d)	$1$2$3$4$5$6
meow	hello
meetup	hello
Launceston	*lowercase*
Hobart	*lowercase*
Cradle Coast	*lowercase*

Table A.2: C-MCRDR CA system text-to-speech corrections

Text	Correction
KIT(\d)	K I T $1
KNE(\d)	K N E $1
ICT	I C T
KNE	K N E
etc	etcetera
[Ll]aunceston	loncesston
\n	\.
(\d)-(\d)	$1 to $2
utas.edu.au	you taz dot e d u dot ay you
KIT	K I T
hr/wk	hours per week
kne(\d)	K N E $1
(.*) *([ap]m) *- *(.*) *([ap]m)	$1$2 to $3$4
(.*)(:00) *- *(.*) *(:00)	$1$2 to $3$4
click above to show formatted data	My response has formatted data that I cannot speak.
printf	print eff
scanf	scan eff
hrs/wk	hours per week
wrt	with respect to
HBT	Hobart
LTN	loncesston
NWC	Cradle Coast
[cC][eE][nN][tT] (\d)	Centenary $1
V(\d)	Building V $1
Dermoudy	dermeh dee
Wassinger	woss ing ger
Saurabh	sore rub
Xu	shoe
Shuxiang	shoe she ang
[cC][eE][nN][tT](\d)	Centenary $1
topic(s)	topic or topics
exam	ex am
day:1	Monday
day:2	Tuesday
day:3	Wednesday
day:4	Thursday
day:5	Friday
class type:Lecture	Lecture
class type:Tutorial	Tutorial
class type:Practical	Practical
class type:Workshop	Workshop
:09:00:00	nine a. m.
:10:00:00	ten a. m.
:11:00:00	eleven a. m.
:12:00:00	twelve p. m.
:13:00:00	one p. m.
:14:00:00	two p. m.

Continued on next page

Table A.2 – *Continued from previous page*

Text	Correction
:15:00:00	three p. m.
:16:00:00	four p. m.
:17:00:00	five p. m.
:18:00:00	six p. m.
:09:50:00	ten to ten a. m.
:10:50:00	ten to eleven a. m.
:11:50:00	ten to 12 p. m.
:12:50:00	ten to one p. m.
:13:50:00	ten to two p. m.
:14:50:00	ten to three p. m.
:15:50:00	ten to four p. m.
:16:50:00	ten to five p. m.
:17:50:00	ten to six p. m.
:18:50:00	ten to seven p. m.
:19:50:00	ten to eight p. m.
(\d)(\d)(\d)(\d)(\d)(\d)	$1 $2 $3 $4 $5 $6
e.g.	for example
Byeong	Bye yong
Huang	Hwang

A.2 C-MCRDR CA System Knowledge-base

Note for space reasons the C-MCRDR knowledge-base tree is represented in Figure A.1 with rules that share the same parent grouped into stacks and/or the same rectangular representation.

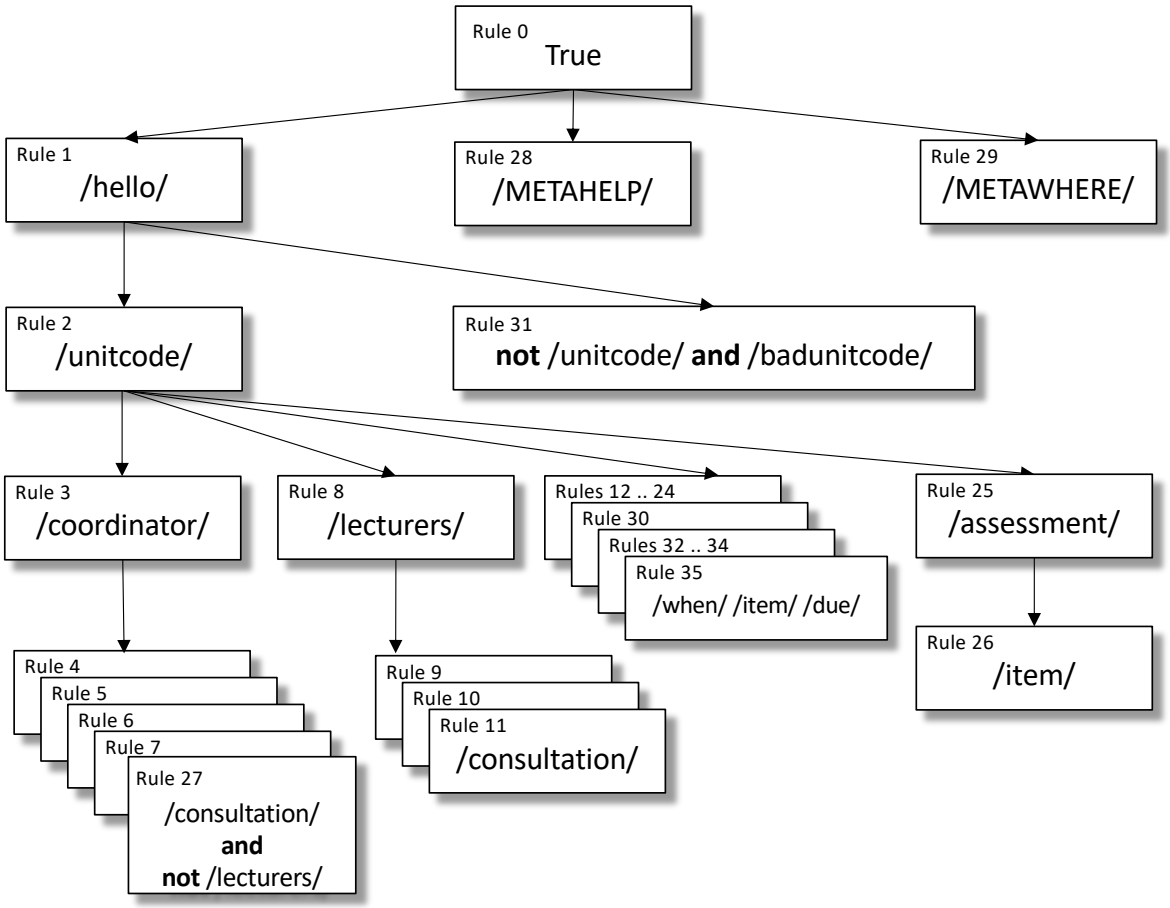


Figure A.1: C-MCRDR CA system knowledge-base tree

Table A.3: C-MCRDR CA system knowledge-base rules

Rule	Parent	Condition	Conclusion
1	0	contains exact term /hello/	I can answer questions about unit outlines for ICT during <LIT>@SYSTEMsemester</LIT>, <LIT>@SYSTEMyear</LIT>. Please specify a unit code..
2	1	contains exact term /unitcode/	OK, we're now discussing <QREF>5<DESC>Specific unit code</DESC></QREF>. What would you like to know?
3	2	contains exact term /coordinator/	The unit coordinator for <LIT>unitcode</LIT> is <QREF>6<DESC>Unit Coordinator</DESC></QREF>.
4	3	contains exact term /contact/	The contact details for <LIT>unitcode</LIT>'s unit coordinator are \n<QREF>7<DESC>Unit coordinator contact details</DESC></QREF>.
5	3	contains exact term /email/	<QREF>6<DESC>Unit Coordinator</DESC></QREF>'s email address is <QREF>8<DESC>EMAIL unit coordinator</DESC></QREF>.
6	3	contains exact term /phone/	<QREF>6<DESC>Unit Coordinator</DESC></QREF>'s phone number is <QREF>9<DESC>PHONE unit coordinator</DESC></QREF>.
7	3	contains exact term /location/	<QREF>6<DESC>Unit Coordinator</DESC></QREF> is primarily based in the following location - \n<QREF>10<DESC>LOCATION unit coordinator</DESC></QREF>.
27	3	contains /consultation/ and not contain /lecturers/	Consultation times are normally associated with the lecturers for a unit - the unit coordinator may not actually lecture in some circumstances. Please ask about the lecturing staff consultation times instead.
8	2	contains exact term /lecturers/	<REPLACE> Lecturing staff for <LIT>unitcode</LIT> include the following people - <QREF>11<DESC>staff for specific unit</DESC></QREF> <REPLACE-EMPTY> Unfortunately I do not know who the lecturers are for this unit. <REPLACE-OVERRIDE> </REPLACE>.

Continued on next page

Table A.3 – *Continued from previous page*

Rule	Parent	Condition	Conclusion
9	8	contains exact term /contact/	<REPLACE> Contact details for <LIT>unitcode</LIT>'s lecturing staff are the following -\n<QREF>12<DESC>lecturer contact details</DESC></QREF> <REPLACE-EMPTY> The lecturing staff for this unit aren't specified, so I don't know their contact details. <REPLACE-OVERRIDE> </REPLACE>.
10	8	contains exact term /location/	<REPLACE> You can find the lecturers for <LIT>unitcode</LIT> at the following locations - \n<QREF>14<DESC>lecturer locations</DESC></QREF> <REPLACE-EMPTY> Sorry, but the lecturers for this unit aren't specified, so I don't know where they are.. <REPLACE-OVERRIDE> </REPLACE>
11	8	contains exact term /consultation/	<REPLACE> Consultation details for lecturing staff in <LIT>unitcode</LIT> - \n <QREF>15<DESC>lecturer consultation times</DESC></QREF> <REPLACE-EMPTY> I don't know who the lecturers are, so as a consequence I'm sorry but I can't tell you their consultation details. <REPLACE-OVERRIDE> </REPLACE>.
12	2	contains exact term /introduction/	<LIT>unitcode</LIT> Introduction\n<QREF>16<DESC>unit introduction</DESC></QREF>.
13	2	contains exact term /prerequisites/	<REPLACE> Prerequisites for <LIT>unitcode</LIT> are <QREF>17<DESC>unit prereqs</DESC></QREF> <REPLACE-EMPTY> The unit coordinator (<QREF>6<DESC>Unit Coordinator</DESC></QREF>) has not specified any prerequisites for this unit. That may also mean there are none. <REPLACE-OVERRIDE> </REPLACE>.
14	2	contains exact term /weight/	The weighting of <LIT>unitcode</LIT> is <QREF>18<DESC>unit weight</DESC></QREF>.

Continued on next page

Table A.3 – *Continued from previous page*

Rule	Parent	Condition	Conclusion
15	2	contains exact term /attendance/	The attendance/performance requirements and teaching and learning strategies for <QREF>5<DESC>Specific unit code</DESC></QREF> are - \n <QREF>19<DESC>unit attendance</DESC></QREF>.
16	2	contains exact term /communication/	<REPLACE> <LIT>unitcode</LIT> communication details - \n<QREF>20<DESC>unit communication</DESC></QREF> <REPLACE-EMPTY> The unit coordinator (<QREF>6<DESC>Unit Coordinator</DESC></QREF>) hasn't specified any communication details for this unit. <REPLACE-OVERRIDE> </REPLACE>.
17	2	contains exact term /teaching/	Classes for <LIT>unitcode</LIT> in <LIT>@SYSTEMsemester</LIT> are - \n<QREF>21<DESC>unit teaching pattern</DESC></QREF>.
18	2	contains exact term /content/	Content for <LIT>unitcode</LIT> is specified as - \n<QREF>22<DESC>unit content</DESC></QREF>.
19	2	contains exact term /learningoutcomes/	<REPLACE> <QREF>5<DESC>Specific unit code</DESC></QREF>'s Learning Outcomes - \n<QREF>23<DESC>unit learning outcomes</DESC></QREF><REPLACE-EMPTY> The unit coordinator (<QREF>23<DESC>Unit Coordinator</DESC></QREF>) has not specified any learning outcomes. <REPLACE-OVERRIDE> </REPLACE>.
20	2	contains exact term /alterations/	<QREF>5<DESC>Specific unit code</DESC></QREF> has been modified based on student feedback in eVALUate surveys since the last offering in the following way(s) - \n\n<QREF>24<DESC>unit alterations</DESC></QREF>.

Continued on next page

Table A.3 – *Continued from previous page*

Rule	Parent	Condition	Conclusion
21	2	contains exact term /prescribed/	<REPLACE><LIT>unitcode</LIT>'s Unit Coordinator <QREF>6<DESC>Unit Coordinator</DESC></QREF> specified the following prescribed text(s) - \n\n <QREF>25<DESC>unit prescribed text</DESC></QREF> <REPLACE-EMPTY> The unit coordinator, <QREF>6<DESC>Unit Coordinator</DESC></QREF>, didn't specify any prescribed texts. <REPLACE-OVERRIDE> </REPLACE>.
22	2	contains exact term /reading/	<REPLACE> Recommended Readings for <LIT>unitcode</LIT> - \n<QREF>26<DESC>unit reading list</DESC></QREF> <REPLACE-EMPTY> There are no recommended readings specified, but check with the unit coordinator <a href="mailto:<QREF>8<DESC>EMAIL unit coordina- tor</DESC></QREF>?subject=Recommend Readings for <LIT>unitcode</LIT>"><QREF>6<DESC>Unit Coordinator</DESC></QREF> for further information. <REPLACE-OVERRIDE> </REPLACE>.
23	2	contains exact term /software/	Specific software being used in <QREF>5<DESC>Specific unit code</DESC></QREF> includes - \n<QREF>27<DESC>unit software</DESC></QREF>.
24	2	contains exact term /assessmentpattern/	The overall assessment pattern for <LIT>unitcode</LIT> is defined as <QREF>28<DESC>unit assessment pattern</DESC></QREF>.
25	2	contains exact term /assessment/	I am aware of <QREF>30<DESC>Count of assessment items for unit</DESC></QREF> assessment items for <LIT>unitcode</LIT>. A summary of each item follows - \n<QREF>29<DESC>unit assessment items</DESC></QREF>. \n\nYou can get more information about a particular item by specifying its number e.g. 'item 1'.

Continued on next page

Table A.3 – *Continued from previous page*

Rule	Parent	Condition	Conclusion
26	25	contains exact term /item/	<LIT>unitcode</LIT>'s <QREF>31<DESC>specific assessment item</DESC></QREF> has the following details - \n\n<QREF>32<DESC>specific assessment item details</DESC></QREF>.
30	2	contains /whatunitcode/	We are currently discussing '<QREF>5<DESC>Specific unit code</DESC></QREF>'.
32	2	contains /isthere/ /exam/	<REPLACE> <QREF>43<DESC>Unit has formal exam</DESC></QREF> <REPLACE-EMPTY> As far as I can tell, there is no formal exam. <REPLACE-OVERRIDE> Yes, there is an exam. </REPLACE>
33	2	contains /isthere/ /groupwork/	<REPLACE> <QREF>44<DESC>Unit has group work</DESC></QREF> <REPLACE-EMPTY> No, There are no group-based assessment items. <REPLACE-OVERRIDE> Yes, one or more assessment items are grouped-based. </REPLACE>
34	2	contains /priorknowledge/	<REPLACE> <QREF>46<DESC>Prior Knowledge</DESC></QREF> <REPLACE-EMPTY> No prior knowledge or skills have been specified <REPLACE-OVERRIDE> </REPLACE>
35	2	contains /when/ /item/ /due/	<REPLACE> <QREF>31<DESC>specific assessment item</DESC></QREF> has a due date specified to be: <QREF>47<DESC>Due date for specific item number</DESC></QREF> <REPLACE-EMPTY> No due date is specified for this item <REPLACE-OVERRIDE> </REPLACE>
31	1	contains /badunitcode/ and not contain /unitcode/	I'm sorry but I don't currently have any details for that unit. Please try another unit. <METANOHELP>
28	0	contains exact term /METAHELP/	<METAHELP>
29	0	contains exact term /METAWHERE/	<METAWHERE>

Table A.4: C-MCRDR CA system lexical paraphrases

Term	Matches
/phone/	phone, phone number, telephone
/email/	email, email address
/lecturers/	lecturer, lecturers, staff, teacher, teachers, teaching staff
/introduction/	intro, introduction, outline introduction, overview, unit introduction
/communication/	announcements, communication, news
/alterations/	alterations, changed, changes, feedback, modifications, student feedback, what has changed
/prescribed/	book, mandatory book, mandatory text, prescribed, prescribed text, text
/reading/	reading, readings, suggested reading, suggested readings
/software/	applications, programs, software, software applications, software programs
/assessment/	assess, assessment, assessment summary
/exam/	exam, examination
/due/	date, due, due date
/grade/	final grade, grade
/website/	unit web site, unit website, web site, website
/facilities/	computers, facilities, macs, pcs
/ethics/	drinking, eating, ethics, ethics guidelines, facilities usage, guidelines, smoking
/strategies/	assistance, learning strategies, strategies
/ohs/	hazards, occupational health and safety, ohs, risks, safety, whs, work health and safety
/phs/	health
/approach/	approach, approach to learning, code of conduct, conduct
/expectations/	expectations, student expectations

Continued on next page

Table A.4 – *Continued from previous page*

Term	Matches
/plagiarism/	cheating, citing, copying, honesty, plagiarism
/misconduct/	Ordinance of Student Discipline, academic misconduct, misconduct, ordinance
/referencing/	bibliography, endnote, harvard, referencing
/submissions/	cover sheet, coversheet, dropbox, electronic submissions, paper, paper submissions, submissions
/extensions/	extensions, late
/review/	review, review of assessment
/complaints/	complaints, complaints procedure
/attendance/	attend, attendance, attendance pattern, attendance requirements, learning strategies, performance, teaching and learning strategies
/learningoutcomes/	consequences, learning, learning outcomes, objectives, outcomes
/assessmentpattern/	assessment pattern, assessment weighting, pattern
/schoolwebsite/	school web site, school website
/schoolhelpdesk/	help desk, school help desk
/furtherinfo/	further assistance, further help, further info, further information
/facultywebsite/	faculty web site, faculty website
/weight/	unit weight, unit weightings
/worth/	weight, worth
/submit/	hand in, submission, submit
/time/	how long, length, time
/howmany/	count, how many, number
/hello/	hello, howdy, hullo, ciao, sup, hi
/consultation/	consult details, consult times, consultation, appointment, consultation times
/concerns/	concerns

Continued on next page

Table A.4 – *Continued from previous page*

Term	Matches
/METAWHERE/	what are we talking about, where am i
/METAHELP/	help, what can i say, how do i use this
/whatunitcode/	what unit code, which unit, current unit, what unitcode
/what/	which, what
/when/	when is, when
/coordinator/	unit coordinator, administrator, main lecturer, coordinator, supervisor, organiser, organizer, advisor, unicorn, leader, coordinator's
/groupwork/	group work, group-work, group based work, group-based work, group assignments, group-based assignments, group based assessment, group-based assessment
/isthere/	does it have, is there, is there an, is there a, does it have a, does it have an
/item/	item 1, item 2, item 3, item 4, item 5, item 6, item 7, item 8, item 9, assignment 1, assignment 2, assignment 3, assignment 4, assignment 5, assignment 6, assignment 7, assignment 8, assignment 9
/badunitcode/	/re:[k][i][t]\d{3}, /re:[k][n][e]\d{3}
/priorknowledge/	prior knowledge, prior skills, knowledge, skills
/location/	office location, where are they, building, location, campus, office, room, where do i find
/content/	what do I learn, unit content, content, schedule
/contact/	contact details, contact information, how do i contact
/prerequisites/	prerequisites, prereqs, prereq, prerequisite
/requesthelp/	requesthelp
/unitcode/	KIT608, KIT106, KIT312, KIT613, KIT204, KIT303, KIT709, KIT107, KIT313, KIT310, KIT101, KIT506, KIT103, KIT104, KIT412, KIT307, KIT201, KIT206, KIT207, KIT302, KIT406, KIT501, KIT708, KIT712, KIT702, KIT402, KIT403, KIT703, KIT713, KIT714, KIT701, KIT401, KIT102, KIT001
/teaching/	do i have to attend classes, how long are tutorials, do i have to attend, how many tutorials, teaching pattern, how many classes, how many hours, contact hours, laboratories, lab classes, practicals, teaching

Table A.5: C-MCRDR CA system query definition XML tags

XML Tag	Description
QUERY	Query definition
SEL	Dynamic field selector
TBL	Table name specifier
FLD	Table field specifier
FMT	Output format specifier
JOI	Table join specifier
JVA	Table join table name specifier
MVA	Table join table name with multi-value field
CRT	Criterion value specifier
CFI	Criterion value - directly specified literal
CCX	Criterion value - value from named context variable value
CNU	Criterion value - named context variable numerical value
CFC	Criterion value - direct literal followed by named context variable value
CFN	Criterion value - direct literal followed by named context variable numerical value
CNT	Aggregate count function
ORD	Aggregate order by function
PRX	Output format prefix specifier
POX	Output format postfix specifier
PAR	Allow partial criterion match

Table A.6: C-MCRDR CA system query XML syntax

Clause	Syntax
query-clause	<QUERY> numeric-literal { *count-clause* }* / *selector-clause* /* { *join-clause* } [*criteria-clause*] { *order-by-clause* } </QUERY>
count-clause	<CNT> table-name-literal *field-name-clause* </CNT>
selector-clause	*dynamic-selector-clause* | *static-selector-clause*
dynamic-selector-clause	<SEL> context-variable-literal </SEL>
static-selector-clause	*table-and-field-clause* | *format-table-and-field-clause*
table-and-field-clause	<TBL> table-name-literal *field-name-clause* </TBL>
format-table-and-field-clause	<FMT> table-name-literal *field-name-clause* *prefix-clause* *postfix-clause* </FMT>
prefix-clause	<PRX> literal-value </PRX>
prefix-clause	<POX> literal-value </POX>

Continued on next page

Table A.6 – *Continued from previous page*

Clause	Syntax
field-name-clause	<FLD> field-name-literal </FLD>
join-clause	<JOI> *left-join-clause* *right-join-clause* </JOI>
left-join-clause	<JVA> table-name-literal *field-name-clause* </JVA>
right-join-clause	*left-join-clause* | *multi-join-clause*
multi-join-clause	<MVA> table-name-literal *field-name-clause* </MVA>
critera-clause	<CRT> table-name-literal *field-name-clause* *criterion-value-clause* <CRT>
criterion-value-clause	*criterion-literal-clause* | *criterion-context-value-clause* | *criterion-numerical-context-value-clause* | *criterion-literal-and-context-value-clause* | *criterion-literal-and-numerical-context-value-clause*
criterion-value-clause	<CFI> literal-value { *partial-match-clause* } </CFI>
criterion-context-value-clause	<CCX> context-variable-name-literal { *partial-match-clause* } </CCX>

Continued on next page

Table A.6 – *Continued from previous page*

Clause	Syntax
criterion-numerical-context-value-clause	<CNU> context-variable-name-literal { *partial-match-clause* } </CNU>
criterion-literal-and-context-value-clause	<CFC> literal-value *criterion-context-value-clause* </CFC>
criterion-literal-and-numerical-context-value-clause	<CFN> literal-value *criterion-numerical-context-value-clause* </CFN>
partial-match-clause	<PAR> Y </PAR>

Key:
{ } - zero or 1 occurrences
/ / - zero or more occurrences
[ ] - 1 or more occurrences
* - at least one of the marked clauses must occur
| - or

Table A.7: C-MCRDR CA system query reference XML tags

XML Tag	Description
QREF	Query ID reference
DESC	Query description

Table A.8: C-MCRDR CA system query reference XML syntax

Clause	Syntax
query-reference-clause	<QREF> numeric-literal *query-reference-description-clause* </QREF>
query-reference-description-clause	<DESC> literal-value </DESC>

Table A.9: C-MCRDR CA system domain queries

ID	Description	Generated XML from GUI
1	Count of ICT units	<QUERY>1<CNT>unit_outline <FLD>unit_code</FLD> </CNT><CRT>unit_outline <FLD>viewable</FLD> <CCX>@SYSTEMviewable</CCX></CRT> <CRT>unit_outline <FLD>study_period</FLD> <CCX>@SYSTEMsemester<PAR>Y</PAR> </CCX></CRT> <CRT>unit_outline <FLD>study_period</FLD> <CCX>@SYSTEMyear<PAR>Y</PAR> </CCX></CRT> <CRT>unit_outline <FLD>unit_code</FLD> <CCX>@SYSTEMICTunitcodePrefix<PAR>Y</PAR> </CCX></CRT> </QUERY>
3	List of ICT units	<QUERY>3<TBL>unit_outline <FLD>unit_code</FLD> </TBL><TBL>unit_outline <FLD>unit_name</FLD> </TBL><ORD>unit_outline <FLD>unit_code</FLD> </ORD><CRT>unit_outline <FLD>viewable</FLD> <CCX>@SYSTEMviewable</CCX></CRT> <CRT>unit_outline <FLD>study_period</FLD> <CCX>@SYSTEMsemester<PAR>Y</PAR> </CCX></CRT> <CRT>unit_outline <FLD>study_period</FLD> <CCX>@SYSTEMyear<PAR>Y</PAR> </CCX></CRT> <CRT>unit_outline <FLD>unit_code</FLD> <CCX>@SYSTEMICTunitcodePrefix<PAR>Y</PAR> </CCX></CRT> </QUERY>

Continued on next page

Table A.9 – *Continued from previous page*

ID	Description	Generated XML from GUI
5	Specific unit code	<QUERY>5<TBL>unit_outline <FLD>unit_code</FLD> </TBL><TBL>unit_outline <FLD>unit_name</FLD> </TBL><CRT>unit_outline <FLD>viewable</FLD> <CCX>@SYSTEMviewable</CCX></CRT> <CRT>unit_outline <FLD>study_period</FLD> <CCX>@SYSTEMsemester<PAR>Y</PAR> </CCX></CRT> <CRT>unit_outline <FLD>study_period</FLD> <CCX>@SYSTEMyear<PAR>Y</PAR> </CCX></CRT> <CRT>unit_outline <FLD>unit_code</FLD> <CCX>unitcode<PAR>Y</PAR> </CCX></CRT> </QUERY>
6	Unit Coordinator	<QUERY>6<TBL>users <FLD>first_name</FLD> </TBL><TBL>users <FLD>last_name</FLD> </TBL><JOI><JVA>users <FLD>user_name</FLD> </JVA><JVA>unit_outline <FLD>user_name</FLD> </JVA></JOI><CRT>unit_outline <FLD>viewable</FLD> <CCX>@SYSTEMviewable</CCX></CRT> <CRT>unit_outline <FLD>study_period</FLD> <CCX>@SYSTEMsemester<PAR>Y</PAR> </CCX></CRT> <CRT>unit_outline <FLD>study_period</FLD> <CCX>@SYSTEMyear<PAR>Y</PAR> </CCX></CRT> <CRT>unit_outline <FLD>unit_code</FLD> <CCX>unitcode<PAR>Y</PAR> </CCX></CRT> </QUERY>

Continued on next page

Table A.9 – *Continued from previous page*

ID	Description	Generated XML from GUI
7	Unit coordinator contact details	<QUERY>7<FMT>users <FLD>first_name</FLD> <PRX> Name - </PRX><POX></POX></FMT> <FMT>users <FLD>last_name</FLD> <PRX></PRX><POX>\n</POX></FMT> <FMT>users <FLD>email</FLD> <PRX>Email - </PRX><POX>\n</POX></FMT> <FMT>users <FLD>ph_ext</FLD> <PRX>Phone - </PRX><POX>\n</POX></FMT> <JOI><JVA>users <FLD>user_name</FLD> </JVA><JVA>unit_outline <FLD>user_name</FLD> </JVA></JOI><CRT>unit_outline <FLD>viewable</FLD> <CCX>@SYSTEMviewable</CCX></CRT> <CRT>unit_outline <FLD>study_period</FLD> <CCX>@SYSTEMsemester<PAR>Y</PAR> </CCX></CRT> <CRT>unit_outline <FLD>study_period</FLD> <CCX>@SYSTEMyear<PAR>Y</PAR> </CCX></CRT> <CRT>unit_outline <FLD>unit_code</FLD> <CCX>unitcode<PAR>Y</PAR> </CCX></CRT> </QUERY>
8	EMAIL unit coordinator	<QUERY>8<TBL>users <FLD>email</FLD> </TBL><JOI><JVA>users <FLD>user_name</FLD> </JVA><JVA>unit_outline <FLD>user_name</FLD> </JVA></JOI><CRT>unit_outline <FLD>viewable</FLD> <CCX>@SYSTEMviewable</CCX></CRT> <CRT>unit_outline <FLD>study_period</FLD> <CCX>@SYSTEMsemester<PAR>Y</PAR> </CCX></CRT> <CRT>unit_outline <FLD>study_period</FLD> <CCX>@SYSTEMyear<PAR>Y</PAR> </CCX></CRT> <CRT>unit_outline <FLD>unit_code</FLD> <CCX>unitcode<PAR>Y</PAR> </CCX></CRT> </QUERY>

Continued on next page

Table A.9 – *Continued from previous page*

ID	Description	Generated XML from GUI
9	PHONE unit coordinator	<QUERY>9<TBL>users <FLD>ph_ext</FLD> </TBL><JOI><JVA>users <FLD>user_name</FLD> </JVA><JVA>unit_outline <FLD>user_name</FLD> </JVA></JOI><CRT>unit_outline <FLD>viewable</FLD> <CCX>@SYSTEMviewable</CCX></CRT> <CRT>unit_outline <FLD>study_period</FLD> <CCX>@SYSTEMsemester<PAR>Y</PAR> </CCX></CRT> <CRT>unit_outline <FLD>study_period</FLD> <CCX>@SYSTEMyear<PAR>Y</PAR> </CCX></CRT> <CRT>unit_outline <FLD>unit_code</FLD> <CCX>unitcode<PAR>Y</PAR> </CCX></CRT> </QUERY>
10	LOCATION unit coordinator	<QUERY>10<FMT>users <FLD>room</FLD> <PRX> Room - </PRX><POX> (Cent - Centenary Building, Eng - Engineering Building, V - Building V)\n</POX></FMT> <FMT>users <FLD>campus</FLD> <PRX>Campus - </PRX><POX> (HBT - Hobart, LTN - Launceston)\n</POX></FMT> <JOI><JVA>users <FLD>user_name</FLD> </JVA><JVA>unit_outline <FLD>user_name</FLD> </JVA></JOI><CRT>unit_outline <FLD>viewable</FLD> <CCX>@SYSTEMviewable</CCX></CRT> <CRT>unit_outline <FLD>study_period</FLD> <CCX>@SYSTEMsemester<PAR>Y</PAR> </CCX></CRT> <CRT>unit_outline <FLD>study_period</FLD> <CCX>@SYSTEMyear<PAR>Y</PAR> </CCX></CRT> <CRT>unit_outline <FLD>unit_code</FLD> <CCX>unitcode<PAR>Y</PAR> </CCX></CRT> </QUERY>

Continued on next page

Table A.9 – *Continued from previous page*

ID	Description	Generated XML from GUI
11	staff for specific unit	<QUERY>11<TBL>users <FLD>first_name</FLD> </TBL><TBL>users <FLD>last_name</FLD> </TBL><JOI><JVA>users <FLD>user_name</FLD> </JVA><MVA>unit_outline <FLD>teaching_staff</FLD> </MVA></JOI><CRT>unit_outline <FLD>viewable</FLD> <CCX>@SYSTEMviewable</CCX></CRT> <CRT>unit_outline <FLD>study_period</FLD> <CCX>@SYSTEMsemester<PAR>Y</PAR> </CCX></CRT> <CRT>unit_outline <FLD>study_period</FLD> <CCX>@SYSTEMyear<PAR>Y</PAR> </CCX></CRT> <CRT>unit_outline <FLD>unit_code</FLD> <CCX>unitcode<PAR>Y</PAR> </CCX></CRT> </QUERY>
12	lecturer contact details	<QUERY>12<FMT>users <FLD>first_name</FLD> <PRX> Name - </PRX><POX></POX></FMT> <FMT>users <FLD>last_name</FLD> <PRX></PRX><POX>\n</POX></FMT> <FMT>users <FLD>email</FLD> <PRX>Email - </PRX><POX>\n</POX></FMT> <FMT>users <FLD>ph_ext</FLD> <PRX>Phone - </PRX><POX>\n</POX></FMT> <JOI><JVA>users <FLD>user_name</FLD> </JVA><MVA>unit_outline <FLD>teaching_staff</FLD> </MVA></JOI><CRT>unit_outline <FLD>viewable</FLD> <CCX>@SYSTEMviewable</CCX></CRT> <CRT>unit_outline <FLD>study_period</FLD> <CCX>@SYSTEMsemester<PAR>Y</PAR> </CCX></CRT> <CRT>unit_outline <FLD>study_period</FLD> <CCX>@SYSTEMyear<PAR>Y</PAR> </CCX></CRT> <CRT>unit_outline <FLD>unit_code</FLD> <CCX>unitcode<PAR>Y</PAR> </CCX></CRT> </QUERY>

Continued on next page

Table A.9 – *Continued from previous page*

ID	Description	Generated XML from GUI
14	lecturer locations	<QUERY>14<TBL>users <FLD>first_name</FLD> </TBL><FMT>users <FLD>last_name</FLD> <PRX></PRX><POX>'s location - \n</POX></FMT> <FMT>users <FLD>room</FLD> <PRX>Room - </PRX><POX>\n</POX></FMT> <FMT>users <FLD>campus</FLD> <PRX>Campus - </PRX><POX>\n</POX></FMT> <JOI><JVA>users <FLD>user_name</FLD> </JVA><MVA>unit_outline <FLD>teaching_staff</FLD> </MVA></JOI><CRT>unit_outline <FLD>viewable</FLD> <CCX>@SYSTEMviewable</CCX></CRT> <CRT>unit_outline <FLD>study_period</FLD> <CCX>@SYSTEMsemester<PAR>Y</PAR> </CCX></CRT> <CRT>unit_outline <FLD>study_period</FLD> <CCX>@SYSTEMyear<PAR>Y</PAR> </CCX></CRT> <CRT>unit_outline <FLD>unit_code</FLD> <CCX>unitcode<PAR>Y</PAR> </CCX></CRT> </QUERY>
15	lecturer consultation times	<QUERY>15<TBL>users <FLD>first_name</FLD> </TBL><FMT>users <FLD>last_name</FLD> <PRX></PRX><POX>'s consultation details are set to the following - \n</POX></FMT> <FMT>users <FLD>consult_times</FLD> <PRX></PRX><POX>\n\n</POX></FMT> <JOI><JVA>users <FLD>user_name</FLD> </JVA><MVA>unit_outline <FLD>teaching_staff</FLD> </MVA></JOI><CRT>unit_outline <FLD>viewable</FLD> <CCX>@SYSTEMviewable</CCX></CRT> <CRT>unit_outline <FLD>study_period</FLD> <CCX>@SYSTEMsemester<PAR>Y</PAR> </CCX></CRT> <CRT>unit_outline <FLD>study_period</FLD> <CCX>@SYSTEMyear<PAR>Y</PAR> </CCX></CRT> <CRT>unit_outline <FLD>unit_code</FLD> <CCX>unitcode<PAR>Y</PAR> </CCX></CRT> </QUERY>

Continued on next page

Table A.9 – *Continued from previous page*

ID	Description	Generated XML from GUI
16	unit introduction	<QUERY>16<TBL>unit_outline <FLD>introduction</FLD> </TBL><CRT>unit_outline <FLD>viewable</FLD> <CCX>@SYSTEMviewable</CCX></CRT> <CRT>unit_outline <FLD>study_period</FLD> <CCX>@SYSTEMsemester<PAR>Y</PAR> </CCX></CRT> <CRT>unit_outline <FLD>study_period</FLD> <CCX>@SYSTEMyear<PAR>Y</PAR> </CCX></CRT> <CRT>unit_outline <FLD>unit_code</FLD> <CCX>unitcode<PAR>Y</PAR> </CCX></CRT> </QUERY>
17	unit prereqs	<QUERY>17<TBL>unit_outline <FLD>prerequs</FLD> </TBL><CRT>unit_outline <FLD>viewable</FLD> <CCX>@SYSTEMviewable</CCX></CRT> <CRT>unit_outline <FLD>study_period</FLD> <CCX>@SYSTEMsemester<PAR>Y</PAR> </CCX></CRT> <CRT>unit_outline <FLD>study_period</FLD> <CCX>@SYSTEMyear<PAR>Y</PAR> </CCX></CRT> <CRT>unit_outline <FLD>unit_code</FLD> <CCX>unitcode<PAR>Y</PAR> </CCX></CRT> </QUERY>
18	unit weight	<QUERY>18<FMT>unit_outline <FLD>unit_weight</FLD> <PRX></PRX><POX>% of one academic year</POX></FMT> <CRT>unit_outline <FLD>viewable</FLD> <CCX>@SYSTEMviewable</CCX></CRT> <CRT>unit_outline <FLD>study_period</FLD> <CCX>@SYSTEMsemester<PAR>Y</PAR> </CCX></CRT> <CRT>unit_outline <FLD>study_period</FLD> <CCX>@SYSTEMyear<PAR>Y</PAR> </CCX></CRT> <CRT>unit_outline <FLD>unit_code</FLD> <CCX>unitcode<PAR>Y</PAR> </CCX></CRT> </QUERY>

Continued on next page

Table A.9 – *Continued from previous page*

ID	Description	Generated XML from GUI
19	unit attendance	<QUERY>19<TBL>unit_outline <FLD>attendance_requirements</FLD> </TBL><CRT>unit_outline <FLD>viewable</FLD> <CCX>@SYSTEMviewable</CCX></CRT> <CRT>unit_outline <FLD>study_period</FLD> <CCX>@SYSTEMsemester<PAR>Y</PAR> </CCX></CRT> <CRT>unit_outline <FLD>study_period</FLD> <CCX>@SYSTEMyear<PAR>Y</PAR> </CCX></CRT> <CRT>unit_outline <FLD>unit_code</FLD> <CCX>unitcode<PAR>Y</PAR> </CCX></CRT> </QUERY>
20	unit communication	<QUERY>20<TBL>unit_outline <FLD>communication</FLD> </TBL><CRT>unit_outline <FLD>viewable</FLD> <CCX>@SYSTEMviewable</CCX></CRT> <CRT>unit_outline <FLD>study_period</FLD> <CCX>@SYSTEMsemester<PAR>Y</PAR> </CCX></CRT> <CRT>unit_outline <FLD>study_period</FLD> <CCX>@SYSTEMyear<PAR>Y</PAR> </CCX></CRT> <CRT>unit_outline <FLD>unit_code</FLD> <CCX>unitcode<PAR>Y</PAR> </CCX></CRT> </QUERY>
21	unit teaching pattern	<QUERY>21<TBL>unit_outline <FLD>teaching_pattern</FLD> </TBL><CRT>unit_outline <FLD>viewable</FLD> <CCX>@SYSTEMviewable</CCX></CRT> <CRT>unit_outline <FLD>study_period</FLD> <CCX>@SYSTEMsemester<PAR>Y</PAR> </CCX></CRT> <CRT>unit_outline <FLD>study_period</FLD> <CCX>@SYSTEMyear<PAR>Y</PAR> </CCX></CRT> <CRT>unit_outline <FLD>unit_code</FLD> <CCX>unitcode<PAR>Y</PAR> </CCX></CRT> </QUERY>

Continued on next page

Table A.9 – *Continued from previous page*

ID	Description	Generated XML from GUI
22	unit content	<QUERY>22<TBL>unit_outline <FLD>content</FLD> </TBL><CRT>unit_outline <FLD>viewable</FLD> <CCX>@SYSTEMviewable</CCX></CRT> <CRT>unit_outline <FLD>study_period</FLD> <CCX>@SYSTEMsemester<PAR>Y</PAR> </CCX></CRT> <CRT>unit_outline <FLD>study_period</FLD> <CCX>@SYSTEMyear<PAR>Y</PAR> </CCX></CRT> <CRT>unit_outline <FLD>unit_code</FLD> <CCX>unitcode<PAR>Y</PAR> </CCX></CRT> </QUERY>
23	unit learning outcomes	<QUERY>23<TBL>unit_outline <FLD>objectives</FLD> </TBL><CRT>unit_outline <FLD>viewable</FLD> <CCX>@SYSTEMviewable</CCX></CRT> <CRT>unit_outline <FLD>study_period</FLD> <CCX>@SYSTEMsemester<PAR>Y</PAR> </CCX></CRT> <CRT>unit_outline <FLD>study_period</FLD> <CCX>@SYSTEMyear<PAR>Y</PAR> </CCX></CRT> <CRT>unit_outline <FLD>unit_code</FLD> <CCX>unitcode<PAR>Y</PAR> </CCX></CRT> </QUERY>
24	unit alterations	<QUERY>24<TBL>unit_outline <FLD>feedback_alterations</FLD> </TBL><CRT>unit_outline <FLD>viewable</FLD> <CCX>@SYSTEMviewable</CCX></CRT> <CRT>unit_outline <FLD>study_period</FLD> <CCX>@SYSTEMsemester<PAR>Y</PAR> </CCX></CRT> <CRT>unit_outline <FLD>study_period</FLD> <CCX>@SYSTEMyear<PAR>Y</PAR> </CCX></CRT> <CRT>unit_outline <FLD>unit_code</FLD> <CCX>unitcode<PAR>Y</PAR> </CCX></CRT> </QUERY>

Continued on next page

Table A.9 – *Continued from previous page*

ID	Description	Generated XML from GUI
25	unit prescribed text	<QUERY>25<TBL>unit_outline <FLD>prescribed_text</FLD> </TBL><CRT>unit_outline <FLD>viewable</FLD> <CCX>@SYSTEMviewable</CCX></CRT> <CRT>unit_outline <FLD>study_period</FLD> <CCX>@SYSTEMsemester<PAR>Y</PAR> </CCX></CRT> <CRT>unit_outline <FLD>study_period</FLD> <CCX>@SYSTEMyear<PAR>Y</PAR> </CCX></CRT> <CRT>unit_outline <FLD>unit_code</FLD> <CCX>unitcode<PAR>Y</PAR> </CCX></CRT> </QUERY>
26	unit reading list	<QUERY>26<TBL>unit_outline <FLD>reading_list</FLD> </TBL><CRT>unit_outline <FLD>viewable</FLD> <CCX>@SYSTEMviewable</CCX></CRT> <CRT>unit_outline <FLD>study_period</FLD> <CCX>@SYSTEMsemester<PAR>Y</PAR> </CCX></CRT> <CRT>unit_outline <FLD>study_period</FLD> <CCX>@SYSTEMyear<PAR>Y</PAR> </CCX></CRT> <CRT>unit_outline <FLD>unit_code</FLD> <CCX>unitcode<PAR>Y</PAR> </CCX></CRT> </QUERY>
27	unit software	<QUERY>27<TBL>unit_outline <FLD>software</FLD> </TBL><CRT>unit_outline <FLD>viewable</FLD> <CCX>@SYSTEMviewable</CCX></CRT> <CRT>unit_outline <FLD>study_period</FLD> <CCX>@SYSTEMsemester<PAR>Y</PAR> </CCX></CRT> <CRT>unit_outline <FLD>study_period</FLD> <CCX>@SYSTEMyear<PAR>Y</PAR> </CCX></CRT> <CRT>unit_outline <FLD>unit_code</FLD> <CCX>unitcode<PAR>Y</PAR> </CCX></CRT> </QUERY>

Continued on next page

Table A.9 – *Continued from previous page*

ID	Description	Generated XML from GUI
28	unit assessment pattern	<QUERY>28<TBL>unit_outline <FLD>assessment_pattern</FLD> </TBL><CRT>unit_outline <FLD>viewable</FLD> <CCX>@SYSTEMviewable</CCX></CRT> <CRT>unit_outline <FLD>study_period</FLD> <CCX>@SYSTEMsemester<PAR>Y</PAR> </CCX></CRT> <CRT>unit_outline <FLD>study_period</FLD> <CCX>@SYSTEMyear<PAR>Y</PAR> </CCX></CRT> <CRT>unit_outline <FLD>unit_code</FLD> <CCX>unitcode<PAR>Y</PAR> </CCX></CRT> </QUERY>
29	unit assessment items	<QUERY>29<FMT>assess_items <FLD>order_num</FLD> <PRX>Item </PRX><POX>,</POX></FMT> <FMT>assess_items <FLD>title</FLD> <PRX></PRX><POX>,</POX></FMT> <FMT>assess_items <FLD>percent_mark</FLD> <PRX></PRX><POX>%</POX></FMT> <ORD>assess_items <FLD>order_num</FLD> </ORD><JOI><JVA>assess_items <FLD>outline_id</FLD> </JVA><JVA>unit_outline <FLD>outline_id</FLD> </JVA></JOI><CRT>unit_outline <FLD>viewable</FLD> <CCX>@SYSTEMviewable</CCX></CRT> <CRT>unit_outline <FLD>study_period</FLD> <CCX>@SYSTEMsemester<PAR>Y</PAR> </CCX></CRT> <CRT>unit_outline <FLD>study_period</FLD> <CCX>@SYSTEMyear<PAR>Y</PAR> </CCX></CRT> <CRT>unit_outline <FLD>unit_code</FLD> <CCX>unitcode<PAR>Y</PAR> </CCX></CRT> </QUERY>

Continued on next page

Table A.9 – *Continued from previous page*

ID	Description	Generated XML from GUI
30	Count of assessment items for unit	<QUERY>30<CNT>assess_items <FLD>assess_item_id</FLD> </CNT><JOI><JVA>assess_items <FLD>outline_id</FLD> </JVA><JVA>unit_outline <FLD>outline_id</FLD> </JVA></JOI><CRT>unit_outline <FLD>viewable</FLD> <CCX>@SYSTEMviewable</CCX></CRT> <CRT>unit_outline <FLD>study_period</FLD> <CCX>@SYSTEMsemester<PAR>Y</PAR> </CCX></CRT> <CRT>unit_outline <FLD>study_period</FLD> <CCX>@SYSTEMyear<PAR>Y</PAR> </CCX></CRT> <CRT>unit_outline <FLD>unit_code</FLD> <CCX>unitcode<PAR>Y</PAR> </CCX></CRT> </QUERY>
31	specific assessment item	<QUERY>31<FMT>assess_items <FLD>order_num</FLD> <PRX>Item </PRX><POX> - </POX></FMT> <TBL>assess_items <FLD>title</FLD> </TBL><JOI><JVA>assess_items <FLD>outline_id</FLD> </JVA><JVA>unit_outline <FLD>outline_id</FLD> </JVA></JOI><CRT>unit_outline <FLD>viewable</FLD> <CCX>@SYSTEMviewable</CCX></CRT> <CRT>unit_outline <FLD>study_period</FLD> <CCX>@SYSTEMsemester<PAR>Y</PAR> </CCX></CRT> <CRT>unit_outline <FLD>study_period</FLD> <CCX>@SYSTEMyear<PAR>Y</PAR> </CCX></CRT> <CRT>unit_outline <FLD>unit_code</FLD> <CCX>unitcode<PAR>Y</PAR> </CCX></CRT> <CRT>assess_items <FLD>order_num</FLD> <CNU>assessitem</CNU></CRT> </QUERY>

Continued on next page

Table A.9 – *Continued from previous page*

ID	Description	Generated XML from GUI
32	specific assessment item details	<QUERY>32<FMT>assess_items <FLD>assess_type</FLD> <PRX> Type - </PRX><POX>\n</POX></FMT> <FMT>assess_items <FLD>percent_mark</FLD> <PRX>Weight - </PRX><POX>%\n</POX></FMT> <FMT>assess_items <FLD>task_length</FLD> <PRX>Task length - </PRX><POX>\n</POX></FMT> <FMT>assess_items <FLD>lo_links</FLD> <PRX>Links to Learning Outcomes - </PRX><POX>\n</POX></FMT> <FMT>assess_items <FLD>due_comment</FLD> <PRX>Due - </PRX><POX>\n</POX></FMT> <FMT>assess_items <FLD>comment</FLD> <PRX>Description - </PRX><POX>\n</POX></FMT> <FMT>assess_items <FLD>submit_help</FLD> <PRX>How to submit - </PRX><POX>\n</POX></FMT> <JOI><JVA>assess_items <FLD>outline_id</FLD> </JVA><JVA>unit_outline <FLD>outline_id</FLD> </JVA></JOI><JOI><JVA>assess_items <FLD>outline_id</FLD> </JVA><JVA>unit_outline <FLD>outline_id</FLD> </JVA></JOI><CRT>unit_outline <FLD>viewable</FLD> <CCX>@SYSTEMviewable</CCX></CRT> <CRT>unit_outline <FLD>study_period</FLD> <CCX>@SYSTEMsemester<PAR>Y</PAR> </CCX></CRT> <CRT>unit_outline <FLD>study_period</FLD> <CCX>@SYSTEMyear<PAR>Y</PAR> </CCX></CRT> <CRT>unit_outline <FLD>unit_code</FLD> <CCX>unitcode<PAR>Y</PAR> </CCX></CRT> <CRT>assess_items <FLD>order_num</FLD> <CNU>assessitem</CNU></CRT> </QUERY>

Continued on next page

Table A.9 – *Continued from previous page*

ID	Description	Generated XML from GUI
33	Matching assignment assessment item	<QUERY>33<TBL>assess_items <FLD>title</FLD> </TBL><JOI><JVA>assess_items <FLD>outline_id</FLD> </JVA><JVA>unit_outline <FLD>outline_id</FLD> </JVA></JOI><CRT>unit_outline <FLD>viewable</FLD> <CCX>@SYSTEMviewable</CCX></CRT> <CRT>unit_outline <FLD>study_period</FLD> <CCX>@SYSTEMsemester<PAR>Y</PAR> </CCX></CRT> <CRT>unit_outline <FLD>study_period</FLD> <CCX>@SYSTEMyear<PAR>Y</PAR> </CCX></CRT> <CRT>unit_outline <FLD>unit_code</FLD> <CCX>unitcode<PAR>Y</PAR> </CCX></CRT> <CRT>assess_items <FLD>title</FLD> <CCX>assignment<PAR>Y</PAR> </CCX></CRT> </QUERY>
34	Matching assignment assessment item due date	<QUERY>34<TBL>assess_items <FLD>due_comment</FLD> </TBL><JOI><JVA>assess_items <FLD>outline_id</FLD> </JVA><JVA>unit_outline <FLD>outline_id</FLD> </JVA></JOI><CRT>unit_outline <FLD>viewable</FLD> <CCX>@SYSTEMviewable</CCX></CRT> <CRT>unit_outline <FLD>study_period</FLD> <CCX>@SYSTEMsemester<PAR>Y</PAR> </CCX></CRT> <CRT>unit_outline <FLD>study_period</FLD> <CCX>@SYSTEMyear<PAR>Y</PAR> </CCX></CRT> <CRT>unit_outline <FLD>unit_code</FLD> <CCX>unitcode<PAR>Y</PAR> </CCX></CRT> <CRT>assess_items <FLD>title</FLD> <CCX>assignment<PAR>Y</PAR> </CCX></CRT> </QUERY>

Continued on next page

Table A.9 – *Continued from previous page*

ID	Description	Generated XML from GUI
35	Matching assignment assessment item weight	<QUERY>35<TBL>assess_items <FLD>percent_mark</FLD> </TBL><JOI><JVA>assess_items <FLD>outline_id</FLD> </JVA><JVA>unit_outline <FLD>outline_id</FLD> </JVA></JOI><CRT>unit_outline <FLD>viewable</FLD> <CCX>@SYSTEMviewable</CCX></CRT> <CRT>unit_outline <FLD>study_period</FLD> <CCX>@SYSTEMsemester<PAR>Y</PAR> </CCX></CRT> <CRT>unit_outline <FLD>study_period</FLD> <CCX>@SYSTEMyear<PAR>Y</PAR> </CCX></CRT> <CRT>unit_outline <FLD>unit_code</FLD> <CCX>unitcode<PAR>Y</PAR> </CCX></CRT> <CRT>assess_items <FLD>title</FLD> <CCX>assignment<PAR>Y</PAR> </CCX></CRT> </QUERY>
36	Matching assignment assessment item submission	<QUERY>36<TBL>assess_items <FLD>submit_help</FLD> </TBL><JOI><JVA>assess_items <FLD>outline_id</FLD> </JVA><JVA>unit_outline <FLD>outline_id</FLD> </JVA></JOI><CRT>unit_outline <FLD>viewable</FLD> <CCX>@SYSTEMviewable</CCX></CRT> <CRT>unit_outline <FLD>study_period</FLD> <CCX>@SYSTEMsemester<PAR>Y</PAR> </CCX></CRT> <CRT>unit_outline <FLD>study_period</FLD> <CCX>@SYSTEMyear<PAR>Y</PAR> </CCX></CRT> <CRT>unit_outline <FLD>unit_code</FLD> <CCX>unitcode<PAR>Y</PAR> </CCX></CRT> <CRT>assess_items <FLD>title</FLD> <CCX>assignment<PAR>Y</PAR> </CCX></CRT> </QUERY>

Continued on next page

Table A.9 – *Continued from previous page*

ID	Description	Generated XML from GUI
37	unit exam length	<QUERY>37<TBL>assess_items <FLD>task_length</FLD> </TBL><JOI><JVA>assess_items <FLD>outline_id</FLD> </JVA><JVA>unit_outline <FLD>outline_id</FLD> </JVA></JOI><CRT>unit_outline <FLD>viewable</FLD> <CCX>@SYSTEMviewable</CCX></CRT> <CRT>unit_outline <FLD>study_period</FLD> <CCX>@SYSTEMsemester<PAR>Y</PAR> </CCX></CRT> <CRT>unit_outline <FLD>study_period</FLD> <CCX>@SYSTEMyear<PAR>Y</PAR> </CCX></CRT> <CRT>unit_outline <FLD>unit_code</FLD> <CCX>unitcode<PAR>Y</PAR> </CCX></CRT> <CRT>assess_items <FLD>title</FLD> <CFI>exam<PAR>Y</PAR> </CFI></CRT> </QUERY>
38	ICT final grade in semester	<QUERY>38<TBL>unit_admin_global <FLD>content_value</FLD> </TBL><CRT>unit_admin_global <FLD>content_key</FLD> <CFI>final_grade_in_semester</CFI></CRT> </QUERY>
39	global final grade in semester	<QUERY>39<SEL>@SYSTEMglobal_content_selector </SEL><CRT>unit_admin_global <FLD>content_key</FLD> <CCX>@SYSTEMcontent_key_final_grade_in_semester <PAR>Y</PAR> </CCX></CRT> </QUERY>
40	global school web site	<QUERY>40<SEL>@SYSTEMglobal_content_selector </SEL><CRT>unit_admin_global <FLD>content_key</FLD> <CCX>@SYSTEMcontent_key_school_web_site</CCX> </CRT> </QUERY>
41	global faculty website	<QUERY>41<TBL>unit_admin_global <FLD>content_value</FLD> </TBL><CRT>unit_admin_global <FLD>content_key</FLD> <CCX>@SYSTEMcontent_key_faculty_web_site</CCX> </CRT> </QUERY>

Continued on next page

Table A.9 – *Continued from previous page*

ID	Description	Generated XML from GUI
42	List of unit codes	<QUERY>42<TBL>unit_outline <FLD>unit_code</FLD> </TBL><ORD>unit_outline <FLD>unit_code</FLD> </ORD><CRT>unit_outline <FLD>viewable</FLD> <CCX>@SYSTEMviewable</CCX></CRT> <CRT>unit_outline <FLD>study_period</FLD> <CCX>@SYSTEMsemester<PAR>Y</PAR> </CCX></CRT> <CRT>unit_outline <FLD>study_period</FLD> <CCX>@SYSTEMyear<PAR>Y</PAR> </CCX></CRT> <CRT>unit_outline <FLD>unit_code</FLD> <CCX>unitcode<PAR>Y</PAR> </CCX></CRT> </QUERY>
43	Unit has formal exam	<QUERY>43<TBL>assess_items <FLD>assess_type</FLD> </TBL><JOI><JVA>assess_items <FLD>outline_id</FLD> </JVA><JVA>unit_outline <FLD>outline_id</FLD> </JVA></JOI><CRT>unit_outline <FLD>viewable</FLD> <CCX>@SYSTEMviewable</CCX></CRT> <CRT>unit_outline <FLD>study_period</FLD> <CCX>@SYSTEMsemester<PAR>Y</PAR> </CCX></CRT> <CRT>unit_outline <FLD>study_period</FLD> <CCX>@SYSTEMyear<PAR>Y</PAR> </CCX></CRT> <CRT>unit_outline <FLD>unit_code</FLD> <CCX>unitcode<PAR>Y</PAR> </CCX></CRT> <CRT>assess_items <FLD>assess_type</FLD> <CFI>Formal Examination<PAR>Y</PAR> </CFI></CRT> </QUERY>

Continued on next page

Table A.9 – *Continued from previous page*

ID	Description	Generated XML from GUI
44	Unit has group work	<QUERY>44<TBL>assess_items <FLD>assess_type</FLD> </TBL><JOI><JVA>assess_items <FLD>outline_id</FLD> </JVA><JVA>unit_outline <FLD>outline_id</FLD> </JVA></JOI><CRT>unit_outline <FLD>viewable</FLD> <CCX>@SYSTEMviewable</CCX></CRT> <CRT>unit_outline <FLD>study_period</FLD> <CCX>@SYSTEMsemester<PAR>Y</PAR> </CCX></CRT> <CRT>unit_outline <FLD>study_period</FLD> <CCX>@SYSTEMyear<PAR>Y</PAR> </CCX></CRT> <CRT>unit_outline <FLD>unit_code</FLD> <CCX>unitcode<PAR>Y</PAR> </CCX></CRT> <CRT>assess_items <FLD>assess_type</FLD> <CFI>In-Semester - group assignment<PAR>Y</PAR> </CFI></CRT> </QUERY>
45	Unit outline_id for current unitcode	<QUERY>45<TBL>unit_outline <FLD>outline_id</FLD> </TBL><CRT>unit_outline <FLD>viewable</FLD> <CCX>@SYSTEMviewable</CCX></CRT> <CRT>unit_outline <FLD>study_period</FLD> <CCX>@SYSTEMsemester<PAR>Y</PAR> </CCX></CRT> <CRT>unit_outline <FLD>study_period</FLD> <CCX>@SYSTEMyear<PAR>Y</PAR> </CCX></CRT> <CRT>unit_outline <FLD>unit_code</FLD> <CCX>unitcode<PAR>Y</PAR> </CCX></CRT> <CRT>unit_outline <FLD>unit_code</FLD> <CCX>unitcode<PAR>Y</PAR> </CCX></CRT> </QUERY>

Continued on next page

Table A.9 – *Continued from previous page*

ID	Description	Generated XML from GUI
46	Prior Knowledge	<QUERY>46<TBL>unit_outline <FLD>prior_learning</FLD> </TBL><CRT>unit_outline <FLD>viewable</FLD> <CCX>@SYSTEMviewable</CCX></CRT> <CRT>unit_outline <FLD>study_period</FLD> <CCX>@SYSTEMsemester<PAR>Y</PAR> </CCX></CRT> <CRT>unit_outline <FLD>study_period</FLD> <CCX>@SYSTEMyear<PAR>Y</PAR> </CCX></CRT> <CRT>unit_outline <FLD>unit_code</FLD> <CCX>unitcode<PAR>Y</PAR> </CCX></CRT> <CRT>unit_outline <FLD>unit_code</FLD> <CCX>unitcode<PAR>Y</PAR> </CCX></CRT> </QUERY>
47	Due date for specific item number	<QUERY>47<TBL>assess_items <FLD>due_comment</FLD> </TBL><JOI><JVA>assess_items <FLD>outline_id</FLD> </JVA><JVA>unit_outline <FLD>outline_id</FLD> </JVA></JOI><CRT>unit_outline <FLD>viewable</FLD> <CCX>@SYSTEMviewable</CCX></CRT> <CRT>unit_outline <FLD>study_period</FLD> <CCX>@SYSTEMsemester<PAR>Y</PAR> </CCX></CRT> <CRT>unit_outline <FLD>study_period</FLD> <CCX>@SYSTEMyear<PAR>Y</PAR> </CCX></CRT> <CRT>unit_outline <FLD>unit_code</FLD> <CCX>unitcode<PAR>Y</PAR> </CCX></CRT> <CRT>unit_outline <FLD>unit_code</FLD> <CCX>unitcode<PAR>Y</PAR> </CCX></CRT> <CRT>assess_items <FLD>order_num</FLD> <CNU>assessitem<PAR>Y</PAR> </CNU></CRT> </QUERY>

Table A.10: C-MCRDR CA system context variable reference XML tags

XML Tag	Description
LIT	Literal value from named context variable
LNU	Numeric value from named context variable

Table A.11: C-MCRDR CA system context variable reference XML syntax

Clause	Syntax
context-reference-clause	*context-value-clause* | *context-numeric-value-clause*
context-value-clause	<LIT> context-variable-name-literal </LIT>
context-numeric-value-clause	<LNU> context-variable-name-literal </LNU>

Key:
| - or

Table A.12: C-MCRDR CA system context variables

Variable Name	Matching Criteria/Value
assessitem	/item/
assignment	/item/
unitcode	/unitcode/
@SYSTEMICTunitcodePrefix	KIT
@SYSTEMcontent_key_extensions	extensions
@SYSTEMcontent_key_final_grade_in_semester	final_grade_in_semester
@SYSTEMcontent_key_school_web_site	school_web_site
@SYSTEMemptyDBResult	No results found in my database..
@SYSTEMsemester	semester 1
@SYSTEMviewable	1
@SYSTEMyear	2017
@SYSTEMglobal_content_selector	<TBL>unit_admin_global <FLD>content_value</FLD> </TBL>

A.3 C-MCRDR CA System Evaluation

Table A.13: C-MCRDR CA system rule feedback scores

	Rule Feedback Score Counts					
R#	**1**	**2**	**3**	**4**	**5**	**Total**
0	17	9	9	0	3	**38**
1	1	0	3	1	7	**12**
2	1	0	4	1	11	**17**
3	0	0	0	0	0	**0**
8	0	1	1	0	7	**9**
9	0	0	0	0	1	**1**
10	0	0	0	0	0	**0**
11	0	0	0	0	1	**1**
13	0	0	0	1	0	**1**
14	0	0	1	0	0	**1**
15	1	0	0	1	2	**4**
16	0	0	0	0	0	**0**
17	1	4	0	1	9	**15**
18	0	0	0	0	0	**0**
19	0	0	0	0	0	**0**
20	0	0	0	0	1	**1**
21	0	0	0	0	2	**2**
22	0	1	0	1	1	**3**
23	0	0	1	0	0	**1**
24	0	0	1	0	1	**2**
25	0	0	2	0	3	**5**
26	0	0	0	0	5	**5**
28	0	0	0	1	0	**1**
30	0	0	0	0	0	**0**
32	0	0	0	1	1	**2**
33	0	0	0	0	2	**2**
34	0	0	0	0	0	**0**
35	0	0	1	1	2	**4**
Total	**21**	**15**	**23**	**9**	**59**	**127**

Table A.14: C-MCRDR CA system rule feedback session scores

		Session Feedback Score Counts					
R#	**ID**	**1**	**2**	**3**	**4**	**5**	**Total**
2	32	0	0	1	0	0	**1**
3	7	2	0	1	0	0	**3**
3	17	0	0	0	0	2	**2**
7	14	0	0	2	1	4	**7**
7	16	0	0	0	0	4	**4**
7	25	0	0	0	0	4	**4**
8	4	1	1	0	2	4	**8**
8	12	1	0	0	0	2	**3**
8	27	0	0	0	0	1	**1**
9	26	0	3	1	1	2	**7**
11	15	1	2	0	0	4	**7**
11	20	0	0	0	0	1	**1**
11	23	0	0	2	1	0	**3**
12	19	1	1	1	0	3	**6**
13	13	0	0	1	0	0	**1**
14	5	6	0	0	0	0	**6**
15	10	0	0	0	2	0	**2**
15	34	6	0	2	0	7	**15**
17	22	2	3	0	1	1	**7**
17	37	0	0	3	1	1	**5**
19	6	0	0	1	0	0	**1**
19	38	1	0	0	0	3	**4**
21	18	0	1	0	0	1	**2**
25	28	0	0	0	0	1	**1**
27	8	0	4	8	0	13	**25**
29	11	0	0	0	0	1	**1**
Total		**21**	**15**	**23**	**9**	**59**	**127**

Table A.15: C-MCRDR CA system Cat1 feedback session scores

		Cat1 Session Feedback Score Counts					
N	**ID**	**1**	**2**	**3**	**4**	**5**	**Total**
2	32	0	0	1	0	0	**1**
3	7	1	0	1	0	0	**2**
3	17	0	0	0	0	2	**2**
7	14	0	0	0	1	4	**5**
7	16	0	0	0	0	4	**4**
7	25	0	0	0	0	3	**3**
8	4	0	0	0	2	4	**6**
8	12	0	0	0	0	1	**1**
8	27	0	0	0	0	1	**1**
9	26	0	1	1	1	2	**5**
11	15	0	0	0	0	4	**4**
11	20	0	0	0	0	1	**1**
11	23	0	0	0	1	0	**1**
12	19	0	0	1	0	2	**3**
15	10	0	0	0	2	0	**2**
15	34	0	0	1	0	7	**8**
17	22	1	0	0	1	1	**3**
17	37	0	0	2	1	1	**4**
19	6	0	0	1	0	0	**1**
19	38	0	0	0	0	1	**1**
21	18	0	1	0	0	1	**2**
25	28	0	0	0	0	1	**1**
27	8	0	0	0	0	13	**13**
29	11	0	0	0	0	1	**1**
Total		**2**	**2**	**8**	**9**	**54**	**75**

Table A.16: C-MCRDR CA System Cat2 feedback session scores

		Cat2 Session Feedback Score Counts					
N	**ID**	**1**	**2**	**3**	**4**	**5**	**Total**
3	7	1	0	0	0	0	**1**
8	12	0	0	0	0	1	**1**
11	15	0	1	0	0	0	**1**
12	19	1	1	0	0	1	**3**
13	13	0	0	1	0	0	**1**
17	22	0	2	0	0	0	**2**
27	8	0	0	5	0	0	**5**
Total		**2**	**4**	**6**	**0**	**2**	**14**

Table A.17: C-MCRDR CA system Cat3 feedback session scores

		Cat3 Session Feedback Score Counts					
N	**ID**	**1**	**2**	**3**	**4**	**5**	**Total**
7	14	0	0	1	0	0	**1**
8	4	1	1	0	0	0	**2**
8	12	1	0	0	0	0	**1**
9	26	0	2	0	0	0	**2**
11	15	0	1	0	0	0	**1**
11	23	0	0	2	0	0	**2**
14	5	1	0	0	0	0	**1**
15	34	2	0	0	0	0	**2**
17	22	1	1	0	0	0	**2**
17	37	0	0	1	0	0	**1**
27	8	0	2	2	0	0	**4**
	Total	**6**	**7**	**6**	**0**	**0**	**19**

Table A.18: C-MCRDR CA system Cat4 feedback session scores

		Cat4 Session Feedback Score Counts					
N	**ID**	**1**	**2**	**3**	**4**	**5**	**Total**
7	14	0	0	1	0	0	**1**
7	25	0	0	0	0	1	**1**
11	15	1	0	0	0	0	**1**
14	5	5	0	0	0	0	**5**
15	34	4	0	1	0	0	**5**
19	38	1	0	0	0	2	**3**
27	8	0	2	1	0	0	**3**
	Total	**11**	**2**	**3**	**0**	**3**	**19**

Table A.19: C-MCRDR CA system sessional inference requests by category

		Category			
N	**ID**	**Cat1**	**Cat2**	**Cat3**	**Cat4**
2	32	2	0	0	0
3	7	2	1	0	0
3	9	2	0	1	0
3	17	3	0	0	0
3	24	2	0	0	1
4	21	3	0	0	1
6	30	3	0	1	2
6	31	4	0	0	2
6	33	3	1	0	2
6	40	4	0	0	2
7	1	3	1	2	1
7	14	5	0	1	1
7	16	6	0	1	0
7	25	4	0	0	3
7	41	5	0	2	0
8	3	1	0	0	7
8	4	6	0	2	0
8	12	4	1	2	1
8	27	5	0	2	1
9	26	6	0	3	0
11	15	5	2	2	2
11	20	4	1	4	2
11	23	6	0	4	1
12	19	6	5	1	0
13	2	6	0	4	3
13	13	9	1	3	0
13	39	5	0	0	8
14	5	1	0	1	12
15	10	10	0	2	3
15	34	8	0	2	5
15	36	6	0	2	7
17	22	5	2	4	6
17	35	10	0	0	7
17	37	11	0	1	5
19	6	6	0	11	2
19	38	9	0	1	9
21	18	9	1	1	10
25	28	17	0	0	8
26	29	14	0	5	7
27	8	15	5	4	3
29	11	17	0	4	8
Total		**252**	**21**	**73**	**132**

A.4 IPA Evaluation

Table A.20: t-tests for recognition by letter count, by human with Google Home

n	$\mu_{\mathbf{no}}(\sigma_{\mathbf{no}})$	$\mu_{\mathbf{yes}}(\sigma_{\mathbf{yes}})$	**t(df)**	**p**
2	636 (1590.89)	400.44 (1084.15)	t(8.27)=0.36	72.75×10^{-2}
3	438.47 (402.2)	423.69 (379.11)	t(52.11)=0.17	86.47×10^{-2}
4	203.41 (102.21)	151.14 (92.84)	t(31.45)=2.16	$\mathbf{3.85 \times 10^{-2}}$ **
5	268.54 (166.31)	309.55 (151.83)	t(35.94)=-1.07	28.96×10^{-2}
6	509.18 (177.51)	439.99 (183.34)	t(12.79)=1.22	24.63×10^{-2}
7	547 (239.37)	519.73 (222.52)	t(5.57)=0.27	79.57×10^{-2}
8	706.6 (375.4)	794.48 (316.2)	t(4.3)=-0.51	63.25×10^{-2}
9	998.38 (419.48)	1059.4 (463.71)	t(8.56)=-0.39	70.52×10^{-2}
10	1962 (192.73)	1416.06 (551.13)	t(5.37)=4.89	$\mathbf{3.70 \times 10^{-3}}$ **
11	2512 (311.13)	2030.77 (802.47)	t(1.29)=2.05	24.20×10^{-2}
12	3169.71 (1944.31)	3362.22 (1379.98)	t(6.46)=-0.26	80.51×10^{-2}
13	4524.5 (1232.68)	3792.88 (1499.01)	t(3.55)=1.14	32.61×10^{-2}

Table A.21: t-tests for recognition by letter count, by human with Amazon Echo

n	$\mu_{\mathbf{no}}(\sigma_{\mathbf{no}})$	$\mu_{\mathbf{yes}}(\sigma_{\mathbf{yes}})$	**t(df)**	**p**
2	536.81 (1392.86)	341.22 (875.04)	t(22.59)=0.43	67.08×10^{-2}
3	457.38 (396.43)	401.12 (374.37)	t(96.17)=0.73	46.83×10^{-2}
4	163.46 (100.71)	162.11 (95.26)	t(77.78)=0.07	94.71×10^{-2}
5	304.13 (164.68)	297.63 (152.31)	t(56.73)=0.19	85.13×10^{-2}
6	419.19 (198.45)	455.15 (179.42)	t(29.28)=-0.75	45.77×10^{-2}
7	468.65 (238.12)	532.17 (218.97)	t(21.89)=-1.02	32.10×10^{-2}
8	884 (204.77)	774.8 (330.95)	t(25.7)=1.67	10.68×10^{-2}
9	1093.72 (422.6)	1038.51 (474.54)	t(58.1)=0.57	56.98×10^{-2}
10	1548.36 (645.79)	1394.94 (508.76)	t(40.7)=1.13	26.58×10^{-2}
11	2121.52 (838.22)	2010.38 (786.19)	t(43.99)=0.6	55.26×10^{-2}
12	3308.93 (1483.41)	3364.22 (1397.42)	t(46.73)=-0.17	86.57×10^{-2}
13	3650.05 (1355.45)	3906.27 (1542.94)	t(39.35)=-0.69	49.53×10^{-2}

Table A.22: t-tests for recognition by letter count, by *Karen* with Google Home

n	$\mu_{\mathbf{no}}(\sigma_{\mathbf{no}})$	$\mu_{\mathbf{yes}}(\sigma_{\mathbf{yes}})$	**t(df)**	**p**
2	418 (1268.91)	504.43 (1218.43)	t(21.18)=-0.17	86.50×10^{-2}
3	502.79 (419.33)	359.19 (338.22)	t(90.37)=1.88	6.40×10^{-2}
4	187.79 (111.96)	144.43 (80.59)	t(70.41)=2.14	$\mathbf{3.58 \times 10^{-2}}$ **
5	276.79 (155.92)	308.63 (155.6)	t(49.16)=-0.92	36.34×10^{-2}
6	446.07 (165.8)	447.85 (186.69)	t(18.8)=-0.04	97.13×10^{-2}
7	453.5 (214.54)	530.63 (223)	t(14.44)=-1.16	26.38×10^{-2}
8	783.55 (305.05)	790.9 (321)	t(12.9)=-0.07	94.14×10^{-2}
9	1107.6 (555.52)	1051.73 (456.37)	t(4.29)=0.22	83.52×10^{-2}
10	1705.33 (577.69)	1411.45 (545.03)	t(9.46)=1.46	17.58×10^{-2}
11	2271.14 (640.33)	2023.02 (808.61)	t(7.52)=0.97	36.28×10^{-2}
12	3655.6 (1847.29)	3332.59 (1399)	t(4.24)=0.39	71.86×10^{-2}
13	5457.33 (989.34)	3684.26 (1440.06)	t(7.13)=4.01	$\mathbf{4.91 \times 10^{-3}}$ **

Table A.23: t-tests for recognition by letter count, by *Karen* with Amazon Echo

n	$\mu_{\mathbf{no}}(\sigma_{\mathbf{no}})$	$\mu_{\mathbf{yes}}(\sigma_{\mathbf{yes}})$	**t(df)**	**p**
2	483.17 (1317.65)	423.29 (992.75)	t(14.62)=0.12	90.38×10^{-2}
3	451.18 (395.65)	335.9 (327.17)	t(34.33)=1.35	18.64×10^{-2}
4	168.49 (107.21)	156.55 (85.59)	t(94.86)=0.62	53.90×10^{-2}
5	295.84 (165.66)	302.75 (148.61)	t(87.29)=-0.22	82.91×10^{-2}
6	440.19 (173.28)	454.44 (193.2)	t(97.92)=-0.39	69.82×10^{-2}
7	510.89 (226.78)	533.19 (219.14)	t(97.26)=-0.5	61.84×10^{-2}
8	794.28 (292.77)	785.9 (343.96)	t(95.56)=0.13	89.59×10^{-2}
9	977.61 (411.14)	1114.95 (487.92)	t(97.49)=-1.53	13.01×10^{-2}
10	1565.4 (541.09)	1320.21 (539.63)	t(97.32)=2.27	$\mathbf{2.56 \times 10^{-2}}$ **
11	2177.48 (835.86)	1903.3 (740.94)	t(96.61)=1.74	8.58×10^{-2}
12	3583.73 (1405.72)	3104.16 (1396.21)	t(97.89)=1.71	9.02×10^{-2}
13	4153.81 (1352.79)	3571.77 (1557.65)	t(68.76)=1.68	9.66×10^{-2}